

TAG USER'S MANUAL

N67 14252

FACILITY FORM 502

(ACCESSION NUMBER)

226

(PAGES)

CR-80845-

(NASA CR OR TMX OR AD NUMBER)

(THRU)

1

(CODE)

10

(CATEGORY)

L
R-847

JPL P.O. DB5-337357

GPO PRICE \$ _____

June 1966

CESTI PRICE(S) \$ _____

Hard copy (HC) 6.00

Microfiche (MF) 1.25

Prepared by

Kenneth O. King

Kenneth Gillett

William Libaw

FORM 502-67

PLANNING RESEARCH CORPORATION
LOS ANGELES, CALIF. WASHINGTON, D.C.

This work was performed for the Jet Propulsion Laboratory,
California Institute of Technology, sponsored by the
National Aeronautics and Space Administration under
Contract NAS7-100.

FOREWORD

The TAG program was written at the Jet Propulsion Laboratory by Mr. William J. Thomas and is based on an approach to network analysis developed by Dr. Kenneth Lock in his doctoral thesis for the California Institute of Technology entitled Coordinate Selection in Numerical Analysis. This manual was written at the end of a year's study and usage of the TAG system. A great deal of the material presented here is a direct result of this year of experience. Equally important sources, however, were the many conversations with the program author and a document written by him entitled Transient Analysis Generator (TAG). This is a first attempt at a detailed, engineering-oriented user's manual and may include a few unintentional errors and unclear explanations. Any areas of questionable accuracy or clarity should be brought to the attention of Mr. William Shubert of the Jet Propulsion Laboratory who acted as responsible engineer over the contract under which this work was financed. It is advised that this document be revised and updated periodically to improve its accuracy and clarity and to keep it up to date with changes in the TAG system.

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ii
I.0 INTRODUCTION TO TAG CIRCUIT ANALYSIS.....	1-1
2.0 TAG SYSTEM DESCRIPTION	2-1
A. Introduction	2-1
1. General Requirements of Circuit Analysis.	2-1
2. General Functional Description of TAG....	2-3
B. Class of Problems Solved by TAG.....	2-5
1. Types of Simulation Performed by TAG.....	2-5
2. Types of Networks Simulated by TAG.....	2-8
C. Operational Description of TAG.....	2-12
D. Limitations to TAG Simulation.....	2-20
1. Network Restrictions Imposed by the Requirements of Realization	2-20
2. TAG System Imposed Restrictions	2-21
3. Accuracy and Economic Restrictions.....	2-24
E. Computing System Requirements for Running TAG.	2-26
3.0 SET UP OF TAG DESCRIPTION AND DATA DECKS.....	3-1
A. Introduction.....	3-1
B. Topological Description.....	3-2
1. General Considerations	3-2
2. Rules for Assigning Node and Transformer Identification Numbers	3-5
3. Rules for Generating Branch Descriptors...	3-7
4. Connection List Format and Punctuation...	3-17
5. Example Connection List Formulation.....	3-19

TABLE OF CONTENTS
(Continued)

	<u>Page</u>
C. Non-Standard Component Description	3-21
1. General Discussion	3-21
2. Multiple Element Devices	3-22
3. Common Properties of Parameter Description Statements	3-24
4. Time Dependent, Parameter Description Statements	3-36
5. Circuit Dependent, Parameter Description Statements	3-47
D. Requirement and Usage of Fortran Specifica- tion Statements	3-65
1. General Considerations	3-65
2. The DIMENSION Statement	3-66
E. The TAG Output and Control Sequence and Description Deck End Statement	3-68
1. General Characteristics of the Output and Control Sequence	3-68
2. Automatic Print-Out Specification	3-73
3. Automatic Plot-Out Specification	3-76
4. The TAG Description Deck END Statement...	3-80
5. Example Output Sequence and END	3-80
F. Set-Up of the TAG Data Deck	3-82
1. General Characteristics of the TAG Data Deck	3-82
2. Data List Format	3-83
3. Multiple Data Lists	3-86

TABLE OF CONTENTS (Continued)

	<u>Page</u>
4. TAG Internal Control Parameters.....	3-87
4.0 TAG SYSTEM DECK SET-UP.....	4-1
A. General Comments.....	4-1
B. Contents and Order of Example TAG System Decks.	4-5
1. General One Pass Preprocessor and Simulation Run.....	4-5
2. TAG Preprocessor Run.....	4-6
3. Fortran Simulation Program Run.....	4-7
4. Binary Simulation Program Run.....	4-8
5.0 EXAMPLE TAG ANALYSES.....	5-1
A. Introduction.....	5-1
B. Example Colpitts Oscillator Simulation.....	5-1
1. Problem Preparation.....	5-1
2. Construction of TAG Equivalent Circuit...	5-3
3. Creation of Coding Sheet Description of TAG.....	5-5
C. Simulation Output.....	5-12
Computer Listing--TAG Circuit Description	
Computer Listing--TAG Generated Solution Program	
Computer Generated Output Curves SC4020 Plotter	
Computer Listing--TAG Circuit Description and Data Decks for Two Cases	
Computer Generated Output Curves SC4020 Plotter	

1.0 INTRODUCTION TO TAG CIRCUIT ANALYSIS

The automatic circuit analysis program described in this manual is called TAG for Transient Analysis Generator. TAG is actually a system of programs which provide a means for simulating both the transient and DC steady state behavior of a large class of electrical networks. The TAG system was designed to provide this capability at a minimum of effort on the part of the user. As a first effort it comes remarkably close to achieving most of its goals. Not only are a very large class of lumped linear, bilateral, passive networks covered by this analyzer but a significant number of non-linear, non-bilateral networks are also capable of being accurately simulated.

TAG is called an analysis generator because it does indeed generate a special analysis program for each circuit described to it. The language by which the circuit topology, parameter values, and type of simulation desired are described to the TAG system are discussed in detail in section 3 of this manual. While an initial perusal of this section might lead to the impression that TAG is difficult to use, a couple of sample problems should be all that is necessary for the average user to achieve a reasonable facility in the detailed procedures required by an average circuit. The simulation program generated by TAG is in the language of Fortran II and may be understood with relative ease by anyone who has access to any of the many available Fortran manuals.

The creation of a special simulation program for each circuit which is in an easily understood and manipulated programming language provides two features that are unique

to the TAG system and give it at least a philosophical advantage over all other known circuit analysis programs. Program readability allows the user to check the actual equations that are generated to simulate the behavior of a particular circuit. These equations appear in symbolic form in matrix format and provide an additional check against errors in the circuit description. Easy manipulation of the simulation program provides unmatched flexibility in the number and type of simulations capable of being performed.

One further feature that increases the usefulness of TAG, over most competitive programs is its capability of handling a wide variety of non-linear component models. Desired model functions may be incorporated into the system by any user in the form of Fortran II arithmetic statements and subroutines. For transient analysis both continuous non-linear and linear-segmented functions may be created within the framework of the system. While, at present, some difficulty arises in the case of non-linear DC steady state analysis, a reasonable number of such circuits have been successfully simulated to show that such analyses can be achieved.

The TAG system is composed of two basic parts. The first part, which actually generates the solution program, is called the generator or preprocessor. The preprocessor interprets the user's circuit description, generates the proper set of simulation equations, and imbeds them in the Fortran solution program. This solution program is unique to the given circuit topology and is available in punched card form as an output of the TAG system. In this form it may be rerun anytime a circuit of that

particular type requires simulation. This program is like any other Fortran II program and may be run without reference to the TAG preprocessor as long as all required subroutine coding is provided during such executions.

The second part of TAG is the execution or simulation system. This system is comprised of the set of subroutines required to actually run a generated solution program. These subroutines provide the detailed solution and control processes required by the simulation program. While the preprocessor is stored on magnetic tape to be called into use by a couple of special control cards, the execution system is stored on cards in relocatable binary form and must be included in all TAG system decks submitted for a simulation run.

Preprocessing and execution may be combined in a single computer run. In such cases it is not necessary to retrieve the solution program in card form unless it is desired for future use. The system, when run in this one pass mode, provides a simplicity of operation that the beginner should take advantage of. The solution program should not be of concern until the user has achieved an adequate facility in the mechanics of setting up problems and running them.

As the user acquires a facility in the basic language and set up procedures of TAG, the great flexibility of the overall TAG system will become of greater and greater use. This flexibility provides for a wide range of analytical investigation and experimentation. Each user will develop a methodology for using TAG and will undoubtedly find areas in which TAG procedures may be improved. This first user's manual will hopefully provide enough material to get people started.

2.0 TAG SYSTEM DESCRIPTION

A. Introduction

1. General Requirements of Circuit Analysis - The general problem of circuit analysis will be defined here as the task of determining, according to some criteria, the degree to which a specific circuit topology performs to a given functional specification. The criteria are often applied in the process of selecting the range of component parameter distributions for which the circuit will be required to function properly. Three of the better known parameter distribution criteria used are nominal, worst-case, and Monte Carlo. However the rules of analysis are established, the basic element in any method is a procedure for relating the established parameter distributions to actual circuit performance. The most direct method of accomplishing this is to build the actual circuit and test it in the laboratory. Unfortunately this method is not only time-consuming and expensive but lacks the facility for complete control over the circuit parameter distributions treated which, in turn, defeats the original intent of the analysis. Automatic circuit simulation by computer represents a second procedure which, at least theoretically, has all the directness of the breadboard and test method but none of its disastrous uncertainty. Ease and speed of simulation represent the two most important features required of any such process.

The TAG program was designed to provide just such a high speed circuit simulator for a reasonably broad class of electronic circuits. With some modification TAG's ease of use and range of circuit capability should be expendable to that covered by any other presently available program. Its unique structure also makes it ideally suitable for further development within the present rapidly developing state of the art of computer programming and hardware.

Because of the wide range of analysis rules for which simulation must be performed, and the wide and ever changing range of electronic component types, program flexibility is an additional important feature of any circuit analysis program. In this area TAG is uniquely superior to any other program known to exist. Component models may be added, modified, or deleted by even the most unsophisticated user. More advanced users can even add extra equations of constraint to the circuit simulation equations to allow the direct computation of performance specification states and eliminate the usual procedure of interpolating such information from performance curves. At the most advanced levels extra programming statements may be added to any individual circuit simulation program, which will automatically provide the extra data manipulation, bookkeeping, and computation required by any of the iterative or statistical analysis techniques.

2. General Functional Description of TAG - The TAG system is most generally viewed as a digital simulator of electronic circuits. The mode of simulation is a set of first order differential and/or algebraic equations which describe the behavior of that set of variables sufficient to fully describe the state of the circuit for any time during the period of simulation. This set of equations is automatically generated by TAG from a description of the circuit topology, and expressed symbolically in an easily understood matrix format. TAG then generates a special Fortran II source program which is tailored to solve the particular set of simulation equations which fit the network under analysis. This special solution program, in which the simulation equations are imbedded, provides all the functions required to describe special component functions, print and plot desired circuit variables, input any desired set of component parameter values and perform the actual solution processes.

The two features of TAG which make it unique among available circuit analysis programs are the generation of simulation equations in symbolic format, and the generation of a solution program tailor-made to solve those equations. The solution program which is available to the user in a permanent form as a Fortran source program card deck is unique to the circuit topology and does not reflect any given set of component values. Therefore a given TAG generated solution program can be used to simulate a given

network for almost any distribution of component parameter values desired (there are certain numerical process limitations to this feature). The source program deck is also available to the user for modification of mathematical component models or addition of program steps to help implement any desired analytical process (i.e., automatic worst case, Monte Carlo, etc.).

The levels of system flexibility described above will be generally useful only to the more advanced users. At the simplest level TAG can be run as a single pass, production program for which the user need not be aware of any of its internal process steps. Simulation of standard, well behaved circuits can be set up and run following a "cook book" set of directions. Unfortunately, some problems still plague the solution processor, and operation at this lowest level is not as effective as it might be. This situation will hopefully be remedied as the TAG program becomes more fully developed and debugged.

In summary, TAG provides a means of rapidly and accurately simulating the behavior of a large class of electronic circuits. In addition, its unique structure makes it flexible enough to be adapted to almost any set of analysis rules desired. Finally, its conceptually advanced mechanization should allow it to grow in capability as the state of the art in this area develops.

B. Class of Problems Solved by TAG

1. Types of Simulation Performed by TAG - In terms of procedural classifications TAG performs both DC steady state and transient analysis of electrical networks. This type of classification, while very meaningful in terms of computational efficiency, tends to be somewhat misleading since the DC and AC steady state behavior can be viewed as the long term transient response of a network to voltage or current excitations which are respectively either constant or sinusoidally varying in time. For this reason and because its behavior is more closely analogous to the actual physical system, transient analysis can be considered to be the most basic of the three processes mentioned. However, there are many important cases for which both DC and AC steady state analysis are much preferred over transient analysis, because of the computational efficiencies which have been developed to handle each process, and because the special situations treated by these two procedures very closely match the true circuit behavior. DC steady state analysis is probably of more immediate importance because of its ability to economically provide initial values to transient analysis runs. AC steady state analysis should, however, be added to TAG in order to complete its repertoire of capabilities.

Transient and DC steady state analysis can be combined to perform the following more general types of analysis.

a. Pure DC Steady-State Analysis

Static operating point analysis: The DC steady-state response to a single set of fixed voltage and current sources is calculated.

Static transfer characteristic analysis: A series of DC steady-state responses is calculated as one of the static voltage or current sources is stepped through a range of values.

Low frequency transient analysis: This is the same as the static transfer characteristic analysis except that values taken on by the varying source are functionally related to the independent variable time. It is assumed here that the time period over which the circuit changes state is very long with respect to all local circuit time constants.

Recurrent low frequency transient and AC steady-state analysis: This is the same as the low frequency transient analysis except that the wave form of the input source is recurrent or sinusoidal.

b. Combined DC Steady State and Transient Analysis

General transient analysis: DC steady state analysis is used to supply the initial conditions to the desired transient performance

run. DC steady state analysis can also be used to eliminate circuit time constants which are too short to be of interest at the desired solution resolution. The time response of interest is generated by integrating the transient simulation differential equations from time t_0 to some future time t_f .

Recurrent transient and AC steady state analysis: This is the same as static operating point analysis except that the input wave form is either generally recurrent in time or specifically sinusoidal. Initial condition transients are allowed to subside.

c. Transient Analysis Only

Power on sequence analysis: Here the initial conditions are always zero and only the initial time response to the application of a voltage or current source is desired. This may be extended to apply to a sequence of power on steps. Even here it may be desirable to eliminate some of the small time constants by using the DC steady-state representation of a portion of the circuit.

While the TAG system may be called upon to perform any or all of the above analyses for a given circuit, the present system may not perform economically. The circuit topology itself

is the biggest and, unfortunately, at present, the least understood factor in determining the economic effectiveness of TAG. Reliable rules of thumb are not yet available to provide guidance to the average engineer. The processes involved are becoming better understood, and such a set of rules should be forthcoming.

2. Types of Networks Simulated by TAG - The TAG program will theoretically generate a reliable mathematical model to represent any physically realizable interconnection of idealized voltage sources, capacitances, conductances, reciprocal inductances, current sources, and ideal transformer windings. However, there obviously exists a set of limiting factors which establish the practical bounds within which TAG's theoretical capability will operate satisfactorily. Since it is intended here to define TAG's capabilities in terms of allowed element type only, these practical system limitations will be treated in detail in a later section. For each of the component or element types recognized by the present TAG system there exists a single characterizing parameter which describes its terminal properties. These are listed below:

Element Types	Characterizing Parameter	Branch Equations
Voltage Source	V	$v_j = V$
Capacitance	C	$i_j = C \frac{dv_j}{dt}$
Conductance	G	$i_j = G \cdot v_j$
Reciprocal Inductance*	L	$i_j = L \int v_j dt = L \cdot \phi_j$
Current Source	I	$i_j = I$
Transformer Winding	N	$\frac{v_j}{N_j} = \text{a constant}$ $\sum_{j=1}^n i_j N_j = 0$ for all n windings (j = 1, 2, ---n) of the transformer

*TAG uses the symbol L for reciprocal inductance. This should be kept in mind throughout the text.

TAG will simulate any system for which an accurate electrical analog can be developed in terms of these allowed branch components. Furthermore, since the characterizing parameter values show up only as numbers which determine the co-efficients of the simulation equations, negative values may be used where desired as long as no mathematical problems (singular matrices, unstable integration, etc.), arise from such action. This level of flexibility should add considerably to the general usefulness of the program. For example, if one can generate an accurate lumped thermal model for a given transistor, its temperature behavior along with its electrical behavior could be relatively easily simulated within a single circuit.

Linear circuits will be defined in the TAG sense as any network constructed from the allowed elements for which the characterizing parameters are all either constant or purely time dependent. TAG quite readily simulates all such circuits which are physically realizable and some few (those having negative component values) which are not. Time dependent component parameters may be described by any functional relationship that can be written within the very broad limitations of Fortran II computer language. Discontinuous time dependent functions can be handled by a special step function subroutine provided as part of the TAG system.

Non-linear circuits are defined as those networks which contain elements whose characterizing parameter value is a function of one or more of the circuit dependent variables. Here again functional dependencies must be described within the syntax allowed by Fortran II. Discontinuous functional relationships are accommodated by the ability of the program to stop the solution at any specified value of a dependent variable function, change co-efficients in any of the computational matrices, and restart as though a new problem has been entered whose initial conditions are the final values of the old problem. Thus a junction diode can either be modeled as a current source whose value changes continuously with the voltage across it according to the classical exponential diode equation, or it can be modeled as the parallel combination of a fixed conductance and current source whose values change only at specified breakpoints. Thus either non-linear or piecewise-linear modeling can be accommodated within TAG.

Of the six basic electrical elements recognized by TAG, the ideal transformer winding is the only one which cannot be controlled as a function of time, because it is treated numerically rather than symbolically during the equation generation process. Dependent variable control is further restricted to capacitances, conductances, reciprocal inductances, and current sources. Exercise of dependent variable control over conductances and reciprocal inductances should be handled with care because of the definition assigned to each by the forms of the TAG

generated equations. This will be treated in more detail in later sections.

C. Operational Description of TAG

In this section TAG circuit simulation is presented as a series of five operations performed by the user. Here, the computing system* and TAG program are treated as a perfect machine to which problems are submitted in proper format and from which answers are retrieved as lists of numbers on plotted curves. For many problems the user need not perform any but these five steps. However, as a user becomes more familiar with the system it will become apparent that a much broader usage may be achieved by a deeper understanding of the processor itself.

Step 1: Problem Definition and Evaluation

This step, required by any analysis procedure, is extremely important to the success of a TAG analysis because of the relatively rigid and complex computing system in which TAG is imbedded. Further, TAG will not economically handle all types of analysis on all circuit configurations. During this step the following questions must be answered satisfactorily before proceeding to the TAG process itself.

- a) What is the exact configuration of the circuit to be analyzed?
- b) Do adequate models exist for all components shown on the circuit schematic?

*The computing system contains the computer proper, all peripheral equipment, and all operators required to transform program decks into output listings or plots.

- c) What functional property of the circuit is to be investigated?
- d) Can TAG properly simulate the required function?
- e) What is the nature of the parameter distribution for which the simulation is to be made?
- f) What circuit variables are required as outputs in order to achieve an effective analysis?

Step 2: Preparation of Circuit Schematic for Input to TAG

During this step the original circuit schematic must be transformed into a format consistent with the special language allowed by the TAG processor. This requires the construction of an equivalent circuit in which all special components are shown as combinations of the six basic TAG allowed elements. In addition, all equivalent circuit nodes are assigned numbers which increase consecutively from zero. Finally, all of the electrical components which appear on this circuit schematic must be relabeled in accordance with the rules for describing TAG circuit elements. This set of procedures translates the users circuit schematic into one which may be entered into the TAG system.

In addition to transforming the circuit schematic into a format acceptable to TAG it is also necessary to generate a set of component parameter values which are consistent with the parameter distribution under investigation and the requirements of TAG. Resistance, inductance, and turns ratio numbers must be converted respectively into conductance, reciprocal inductance, and equivalent turns-per-winding. Special components which are described as dependent equivalent TAG elements may very well be controlled by parameters which are not directly available from measurements or data sheets. These must be calculated and formatted to be consistent with TAG.

Step 3: Translate TAG Equivalent Circuit into a Form Acceptable to the Computer

Since the computer will not accept either pictorial or printed information directly, it is necessary to transfer all circuit information onto punched cards which can be read directly by the computing machine. This is generally a two step process in which the required information is first recorded on special coding sheets whose lines each correspond directly to the character format of a punched card and then transferred to a set of cards by a keypunch machine. Generally the user need only prepare the coding forms since both keypunch

machine and operator are usually supplied by computing system.

The coding sheets can be set up easily using only the cook book procedures presented in the second section of the users manual. In general the coding can be copied directly off the equivalent TAG circuit schematic. However, great care must be taken to follow the given procedures exactly. The computer will interpret only what appears on the coding sheets and is completely confused by any error in punctuation, format, or character designation. TAG has been set up to give as much freedom of expression to the user as possible; however, every character which is presented to the computer is essential in that it forces some sort of action to take place. For this reason following the rules to the letter is of utmost importance.

The exact procedures required for using TAG will be presented in the second half of the users manual in enough detail to allow any reasonably experienced circuit engineer to successfully go through the mechanics of setting up problems for TAG. While not an absolute necessity, it should be extremely helpful for a new user to carefully read a Fortran II programming manual. Not only is the general syntax of TAG taken from Fortran II (except for the topological description)

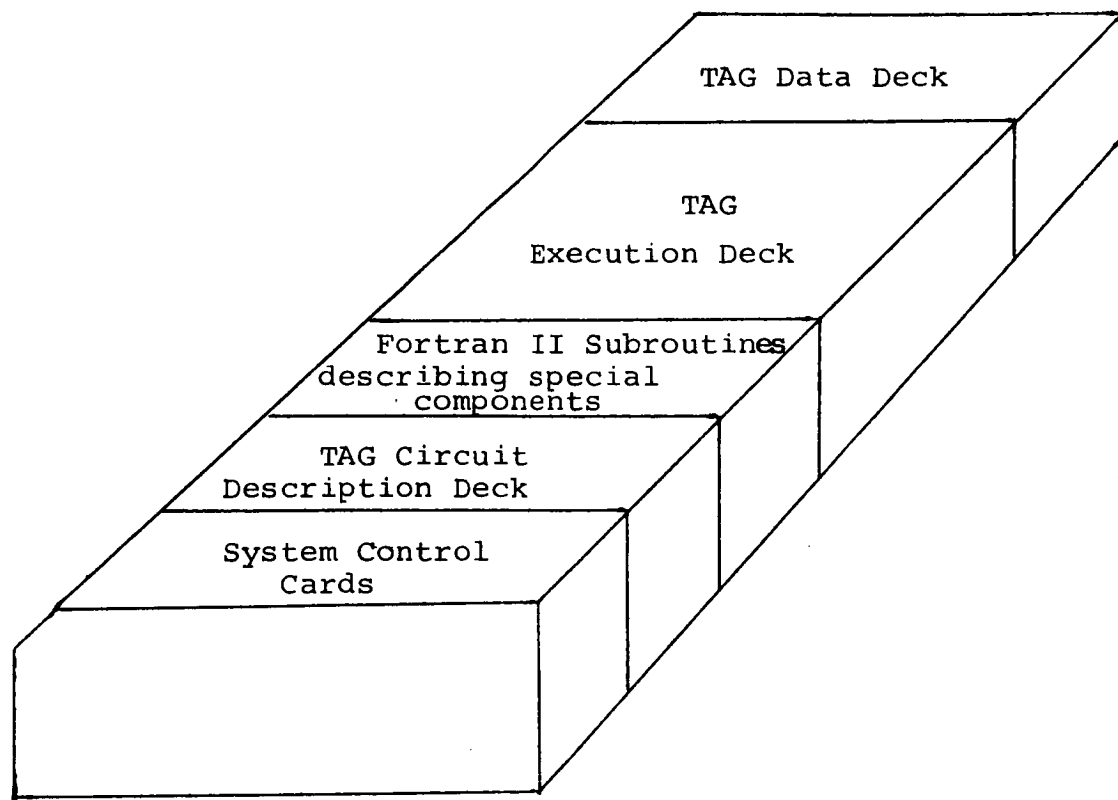
but the Fortran II programming manual contains several helpful discussions which cover computing languages and coding in general.

The deck of cards produced from the coding sheets prepared by the user describes his particular problem in terms of a simulation description deck and a component parameter data deck. The TAG description deck contains all the general information about the circuit topology and type of simulation to be performed. The TAG data deck contains all the information pertinent to the particular component parameter distribution for which the given simulation is desired.

This deck of cards, whether keypunched by the user or by an operator supplied by the computing system, should be rechecked by the user against the original equivalent TAG circuit schematic before proceeding to the next step.

Step 4: Assembly of the TAG System Card Deck and Submittal to the Computing System

In the present JPL system the TAG description and data decks must be combined with certain computing system control cards and a portion of the TAG execution program deck before it can be submitted to the computing system. An outline of the required deck setup is shown on the following page.



System control cards - The system control cards are used to automatically alert the computing system to all the needs of the TAG program. The most important of these sets up the extra magnetic tape units required to run TAG problems and read in the TAG program itself, which is permanently stored on tape, and the Fortran II program, which is permanently stored in the system library.

TAG circuit description deck - As described above this deck contains all the general information pertinent to the circuit topology and type of simulation to be performed.

Fortran II subroutine decks - This deck contains all the user supplied component modeling subroutines. Any non-standard components whose descriptions are too complex to be included in the TAG description deck must be supplied as Fortran II subroutines. These are linked to the description deck by a standard Fortran subroutine call statement which relates the subroutine variables to the TAG equivalent circuit variables.

TAG Execution deck - This deck contains all the subroutines required to execute the TAG produced solution program. These include, for example, the differential equation integration subroutine, FMARK, the non-linear DC steady state solution subroutine, ROOT, the matrix summation, difference and multiplication subroutines, PSUM, MSUM, and MULT, respectively, and many others. These have all been precompiled and appear here in their most compact machine language or binary form.

TAG data deck - The data deck as described above contains all the circuit parameter and simulation control numbers pertinent to a

particular analysis. The first card in the data deck is another special system control card which tells the computing system that the cards which follow are all data values. The last card in the deck is called an "end of file card" and tells the system that the end of the program has been reached.

After the TAG system deck is assembled, as shown, it can be submitted to the system and run. The system operators load the deck into the machine for the run. At run completion they retrieve the deck, the answer listing and plots, and return these to the user.

Step 5: Evaluation of output and rerun capability

The final step, which is as vital to the process of analysis as any other, is a careful evaluation of each simulation. The importance of this step cannot be overstressed. TAG will faithfully simulate any circuit which is properly described to it whether it happens to be the correct one or not. A check on the reasonableness of the answers is always desirable. A second check is provided, however, in the form of the symbolic simulation equations. These are listed in matrix format as part of the Fortran II solution program generated by TAG. With a little practice equation checking is relatively easy to do.

Once a TAG solution program has been generated and proves to be correct for the desired circuit the solution program itself can be generated in card form and submitted in place of the TAG description deck. Removal of certain of the system control cards and a direct substitution of the generated solution source deck for the description deck is all that is necessary. The data deck may be modified at will to allow many component parameter distributions to be run from a single TAG solution program generation run. This can add a great deal to the overall efficiency of TAG analysis.

D. Limitations to TAG Simulation

1. Network Restrictions Imposed by the Requirements of Realization - The practical limitations to circuit size and complexity arise from three separate sources. The first, which is the easiest to understand and cope with, is imposed by the requirement that the network be connectively realizable within the constraints set by the allowed element types.
 - a. There must never be more than one path between any two nodes in a network in which all the branches are voltage sources.
 - b. There must be at least one path between any two nodes in a network in which none of the branches are current sources.

- c. An ideal transformer may have no more than one of its winding voltages fixed by a closed path of voltage sources.
- d. An ideal transformer must have at least one path between the ends of at least one winding which does not contain a current source.
- e. Use of negative element values where simulation requires it must be handled with care since the rules for the realization of such networks are not easily specified.

2. TAG System Imposed Restrictions - There are constraints imposed upon the range of allowed network topologies by the particular mechanization of the TAG processor. Most of the limitations which fall into this group can be reasonably well defined and understood. However, the list given below is probably not complete since many of these limitations have been found in the process of using the program, and it cannot be very confidently assumed that more will not be found in the future.

- a. The present TAG input language has no provision for describing circuits which contain over one-hundred nodes.
- b. No two nodes within a network may be connected by more than one element of the same type.

- c. Voltage step excitation should never be applied to a circuit input port across which there exists any path of all capacitive elements.
- d. Current step excitation should never be applied to a circuit input port across which there is not at least one path in which no inductive elements exist.
- e. Inductor initial condition current must be handled by adding a current source element which parallels the inductor and has a magnitude and direction equal the desired initial current.
- f. Initial condition voltages can be supplied to capacitive branches only. For non-linear DC steady state simulation this restriction, if not circumvented at the solution program level, will often prevent the attainment of a final solution.
- g. The form in which TAG writes the simulation equations implies the following definitions for C, G, and L. In the linear case the given definitions are perfectly adequate; however, the non-linear extensions of the definitions of G and L are of questionable value. Each definition is derived from the method by which TAG calculates the branch currents used in the generalized node equilibrium equations.

$$i_C = C \frac{dv}{dt} \therefore C = \frac{i_e dt}{dv} = \frac{dq}{dv} \quad - \text{ incremental capacitance}$$

$$i_G = G \cdot v \therefore G = \frac{i_G}{v} \quad - \text{ total conductance}$$

$$i_L = L \int v dt \therefore L = \frac{i_L}{\int v dt} = \frac{i_L}{\phi} \quad - \text{ total reciprocal inductance}$$

Experience has shown that it is easier to model a non-linear G or L by a current source which is dependent upon either voltage or voltage integral respectively.

- h. Any node whose voltage must be known should be connected to a resistor or capacitor. A non-linear conductance or inductance modeled by a dependent current source should be considered to be a current source in this respect.
- i. Discontinuous functions of the independent variable (time) which are used to control element values must use the special step function subroutine provided to flag the time at which the discontinuity takes place.
- j. Discontinuous functions of any circuit dependent variable (node pair voltage or voltage integral) which are used to control element values must not be used at all in a DC steady state analysis. In a transient analysis such

dependencies must use the dependent variable stop feature built into the integration routine to allow the simulation equations to properly adjust to such discontinuities.

3. Accuracy and Economic Restrictions - The most difficult set of limitations to understand and deal with are those which are imposed on the range of allowed network topologies by considerations of cost and accuracy. Because of the difficulty in explicitly defining these restrictions in terms of actual circuit configurations, only a general discussion of these two subjects can be given here.

DC steady state analysis of circuits which contain elements whose values are non-linearly dependent upon circuit variables is accomplished by means of an iterative solution technique. This technique requires an initial solution estimate which is corrected according to a procedure called the Newton-Raphson process. In the present TAG system this process is not optimally implemented and will not in general achieve a suitable solution for any arbitrary first guess. This can be circumvented by supplying a reasonable first estimate to the equations. Unfortunately this must be accomplished by the addition of coding at the solution program level.

In addition to the above stated problem the accuracy criteria implemented in the Newton-Raphson subroutine is incorrectly applied. This often leads to situations where the process appears to converge

on an incorrect solution, or fails to converge for a set of answers which are well within the accuracy required.

Both of these problems can be circumvented at the solution program level by additional Fortran II coding. However, this is accomplished at the expense of time and confidence. At present there exists no set of rules by which a user can tell by visual inspection whether a given circuit will suffer from such problems or not. It is felt, however, that significant process improvements applied to this problem could achieve a much more effective system.

Transient solutions are achieved by solving sets of differential equations. Solutions appear as a time history of circuit behavior starting at the initial condition state and proceeding until time reaches some predetermined value. A step-by-step integration process is used to calculate the values of all circuit dependent variables at discrete points in time. The computer time required to achieve a solution is directly proportional to the number of computational steps taken between the initial time and the terminating time. Unfortunately, the time between solution points is related very strongly to the accuracy of the solution and cannot therefore be chosen arbitrarily to simply meet visual resolution criteria. The maximum integration time step allowed for a reasonable accuracy is also strongly related to the interaction of the local circuit

time constants. For this reason circuits having local time constants very much shorter than the time period over which the simulation is desired may require an inordinant amount of machine time to achieve a single solution run. To be safe the time step should be maintained at some level below the smallest circuit time constant.

TAG has the theoretical ability to solve combinations of differential and algebraic equations. This should allow the user to eliminate insignificant time constants by "physically" removing small valued capacitors and inductors from the TAG equivalent circuit. However, because of problems presently existing in the DC steady state equation solver and the time consumed in this process, such a procedure has not proven effective.

The small time constant problem is basic to the method used to achieve transient solutions. No universally satisfactory solution to this problem has been devised to date. Certain procedures have been developed to crudely circumvent this problem in specific cases, where this effect was of critical importance. A significant breakthrough is required in this area before truly universal automatic transient analysis can be achieved.

E. Computing System Requirements for Running TAG

The TAG program is setup at JPL to run on a direct coupled 7040/7094 computing system. The 7040 provides all the input/output functions to the 7094 and schedules

the running of programs on the 7094. This system contains a full set of IBM system subroutines and more than enough extra tape units and disk storage to support a program considerably larger than TAG. TAG itself is permanently stored on tape and automatically read into the system when required. Plotting is provided by a special program which automatically scales, formats, and writes the required data onto a tape for the off-line Stromberg-Carlson SC4020 plotter.

The JPL system is considerably more complex than need be to run TAG. However, for such a large central computing system it does provide extremely good turnaround time (often as low as .5 hours) and a fairly reasonable interface for outside users. Because such a system may not always be available, it is of interest to describe the minimum system required by TAG.

TAG's most basic requirements are an IBM 7094 computer and a Fortran II, MOD III programming system. The generator portion of TAG, which derives the simulation equations and generates the solution program, requires, in addition to normal input/output/punch files, four scratch tapes or equivalent disk file logical units. The execution portion of TAG, which actually performs the simulation, may be run using only the basic system. However, the JPL automatic plot routine which automatically scales, formats and writes the required output data on a special SC4020 plot tape requires, in addition to the output tape mentioned, three scratch tapes and a host of special subroutines which are part of the JPL systems library.

3.0 SET UP OF TAG DESCRIPTION AND DATA DECKS

A. Introduction

This section describes in detail the steps which must be performed by the user in describing any given circuit simulation problem to the TAG system. The description is presented to the computing system in the form of a deck of key-punched IBM cards. The user need create only two relatively small portions of this system deck called the TAG description deck and TAG data deck. The user produces the information for these decks on hand-written IBM coding sheets which may be transformed to punched card form by the user or, more efficiently, by a key punch operator supplied by the computing system.

There are four distinct parts to the TAG description deck. The first is the topological description or connection list which describes the circuit configuration to the TAG processor. The second is the non-standard component description list which describes the functional dependency of any element value which does not remain constant throughout a given simulation run. The third is the Fortran II specification statement list which provides certain information to the Fortran solution program necessitated by certain operations called for in either the non-standard component description list or the output and control sequence. The final section is the output and control sequence which provides the user with the results of a simulation run, controls the overall simulation process, and formally ends the TAG description deck.

The TAG data deck provides the solution program with the actual numerical values of all circuit parameters, initial conditions, and process control parameters required for any particular simulation run. The highly flexible format allowed for listing this information is presented as the final portion of this section.

B. Topological Description

1. General Considerations

The first step required in the application of TAG circuit analysis is the transformation of the original schematic of the given circuit to a TAG equivalent circuit. This process creates a circuit diagram from which the TAG input connection list can be directly generated and which can easily be compared with the original circuit schematic. Once this equivalent schematic is prepared and checked it becomes the official circuit description as far as TAG performance simulation is concerned. The three objectives to be accomplished during the creation of the TAG equivalent circuit are outlined below in their proper order of accomplishment.

a. Circuit Schematic Preparation

Since TAG accepts only circuits which are fully connected, portions of circuitry which appear on the schematic as physically isolated subcircuits must either be analysed separately or connected together physically before analysis. Flux coupling from one transformer winding to another does not properly serve as a physical connection.

For truly isolated circuits it is probably best to treat them separately. However, transformer coupled subcircuits which contain no physical interconnections may be arbitrarily joined together by a common reference node without loss of reality as long as all voltage calculations are made within a sub-circuit.

To achieve satisfactory results at a reasonable cost, circuits under analysis should always be reduced to the lowest level of complexity consistent with the proper simulation of the performance characteristic under investigation. As is true with any analytical device, TAG operates most satisfactorily on easy problems. Superfluous components and nodes should be eliminated as early in the process as possible.

b. Modeling Non-Standard Components in Terms of Standard TAG Elements.

Since TAG recognizes only six basic components - voltage sources, capacitances, conductances, reciprocal inductances, ideal transformer windings, and current sources - the TAG equivalent circuit may contain only these elements. This requires that any component shown on the original schematic that can not be identified as one of these six must be modeled as some interconnected combination of them in order for TAG to create a proper simulation for the given circuit.

c. Translation of Circuit Element Identifiers to TAG Nomenclature

The TAG processor receives the equivalent circuit description as a connection list which describes each network branch according to its basic TAG element type and terminal connections. Creation of the TAG branch descriptors, which are the elements of the connection list, requires the user to establish a numerical identification for each branch interconnection point or node within the TAG equivalent circuit.

In addition to terminal node identification, descriptors for ideal transformer windings require identification of the transformer on which they are wound. To achieve this the user should assign a unique number, independent of the set of node numbers used, to each transformer appearing in the TAG equivalent circuit.

After establishing a node and transformer identification system consistent with the rules given in the next section, the user should clearly label each branch element of the TAG equivalent circuit with its proper TAG descriptor. Preparation of the coding sheets, which is the first step in translating the circuit information to a form acceptable to the computer, will be greatly facilitated by this procedure. The TAG descriptor language, while easy to learn, is absolutely restrictive and must therefore be learned well and used carefully.

2. Rules for Assigning Node and Transformer Identification Numbers

a. Assignment of Node Identification Numbers

Every node or branch interconnection point which appears in the TAG equivalent circuit must be assigned an identification number. These numbers must start with zero and increase sequentially to $N-1$ for an N node circuit. Because the branch descriptor requires a two digit decimal number to specify each node to which it connects, it is helpful to observe this constraint when assigning numbers to the nodes of the network diagram. Therefore, the numbers should start with 00 and proceed 01, 02, - - - ($N-1$). The largest node number provided for is 99.

b. Assignment of Transformer Identification Numbers

Every ideal transformer appearing in the TAG equivalent circuit must be assigned a unique identification number. Like node identifiers, the transformer identification numbers should always contain two digits to conform to the requirements of the transformer winding descriptor. Unlike node identifiers, transformer identification numbers should start with 01 and increase sequentially to K for a circuit containing K transformers.

c. The Significance of Node 00

The number 00 may be used to specify the circuit reference or ground node. While some users might find this to be a useful part of their own personal TAG procedure, it does not represent a necessary constraint of the TAG system. Even though the internal TAG processor uses node 00 as a common basis reference for some of the temporary circuit matrices formed during the equation generation process, the final set of TAG simulation equations requires no externally fixed reference point. All the required transformations between the simulation equation variables and the output display variables will be properly constructed regardless of the function of the node labeled 00.

d. TAG Equivalent Circuit Modification

Because of the TAG imposed constraint that nodes be numbered sequentially, the addition of nodes to achieve extra refinement in the simulation or the deletion of nodes to simplify a given topology is generally accomplished at the expense of relabeling a large number of branch elements. By properly planning the numbering system so that the largest numbers occur in the area of the circuit which is most likely to change, the number of branch elements requiring relabeling can be kept to a minimum.

Because adding nodes will almost always be easier than deleting them, it is probably best to start with the simplest circuit configuration which will adequately simulate the desired circuit property. Increases in complexity to achieve

higher degrees of solution refinement should be added later. This approach will also tend to produce meaningful data at the lowest cost.

The elimination of a low numbered node by the simple act of "Shorting out" all the branches between two nodes, may be achieved by adding a voltage source element between the two nodes to be merged and assigned it a value of zero. This will automatically modify the simulation equations to account for the absolute dependency between the two node voltages and require no changes in the original element descriptors.

3. Rules for Generating Branch Descriptors

Every element which appears as a branch in the TAG equivalent circuit must be one of the six types allowed by the TAG processor. At the topological level TAG makes no distinction between descriptors of standard elements whose characterizing parameter is a constant and non-standard elements whose characterizing parameters may vary during a simulation run as a function of either an independent or dependent circuit variable. All elements required to describe the particular circuit behavior under investigation must appear as standard TAG elements in the TAG equivalent circuit.

The rules for generating TAG element descriptors are presented in detail in the next two subsections. First, the order and function of each of the characters which make up a descriptor is presented.

Second, an example of each of the six basic element descriptors is given and the properties of each discussed in detail. It is hoped that the presentation of these two redundant but complementary discussions will fully clarify both the descriptor generating process and the physical nature of the TAG elements described.

a. The Structure of TAG Branch Descriptors

The first six characters of all branch descriptors follow the same format. This format which is detailed below specifies the element type, according to Table no. 1, and its terminal node numbers. The ideal transformer winding is the only element requiring a more detailed description than can be specified in a six character format. The transformer winding descriptor must contain the information that relates it to all other windings on the same transformer. In addition, its characterizing parameter, the number of turns per winding, which is used numerically during the equation generation process, must be specified. This is illustrated below.

Table no. 1 Element Type Designation

Voltage Source	-V
Capacitance	-C
Conductance	-G
Reciprocal Inductance	-L
Transformer Winding	-N
Current Source	-I

The first six characters of each branch descriptor follow the format shown below.

Character Number	1	2	3	4	5	6
General Descriptor Structure	S	Element Type	Negative Node no.		Positive Node no.	
Example of Voltage Source	S	V	0	1	0	2

Character no.

Function

- 1 An S is placed in this space for all descriptors and has absolutely no meaning for the user.
- 2 This space always contains an alphabetic character which designates the element type according to the rules given in Table no. 1 above.
- 3 The third character always contains the most significant digit of the negative node number.
- 4 The fourth character always contains the least significant digit of the negative node number.
- 5 The fifth character always contains the most significant digit of the positive node number.
- 6 The sixth character always contains the least significant digit of the positive node number.

The transformer winding descriptor follows the format shown below which differs from other descriptors only after the sixth character.

Character Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Xformer Descriptor Structure	S	N	Negative Node no.		Positive Node no.		/	Xformer no.		-	Number of Turns/Winding					
Example	S	N	0	0	0	1	/	0	1	-	1	0	0			

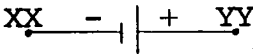

Character no.	Function
1 through 6	Same as above for all other descriptors. The second character is always an N, the general designation for a transformer winding.
7	Character number seven is always a slash (/).
8	The eighth character specifies the most significant digit of the transformer identification number.
9	The ninth character specifies the least significant digit of the transformer identification number.
10	The tenth character is always a dash (-).
11	The eleventh character is the most significant digit of the number of turns per the winding being described.

Character no.

12	The twelfth character is the second most significant digit of the number of turns per the winding being described.
13,14,15,16	The thirteenth through the sixteenth positions are also available for expressing the digits of the number of turns per the winding being described. The maximum number of turns that can be specified for any single winding is 131072. Since the turns always appear in the equations as ratios, the turns numbers for any given transformer may be normalized around any convenient basis. Only the number of characters required to express the number of turns is used; the rest are ignored.

b. TAG Branch Element Description

Voltage Source:

Symbols		Fixed voltage source
		Variable voltage source
Descriptor	SVXXYY	Node order related to polarity
Characteristics	$i_b = ?$ $v_b = V$	Branch current equation Branch voltage constraint

Remarks	The relationship between voltage source polarity, for positive values of source voltage, and the order of the node identification numbers in the voltage source descriptors are shown above. TAG's application of the branch voltage constraint removes the branch current term from all simulation equations. The branch current, therefore, need not be known.
---------	--


Capacitance:

Symbol	$\begin{array}{c} v_{bi} \\ \text{XX} - \text{---} \text{---} + \text{YY} \end{array}$	Initial condition voltage shown
Descriptor	SCXXYY	
Characteristic	$i_b = C \frac{dv_b}{dt}$	Branch current equation

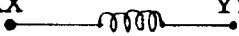
Remarks	Initial condition voltages are entered with the parameter data list rather than the connection list. The initial conditional voltage descriptor is identical to the voltage source descriptor. It must maintain the same node order as the connection list descriptor of the capacitance to which it relates. Polarity is assigned by the sign of the voltage value given the initial conditions descriptor. For the above example the parameter data list must have an entry SVXXYY = (v _{bi}), (where (v) means the numerical value of v).
---------	--

The branch current equation shows that TAG defines capacitance incrementally rather than absolutely. When C is constant the two definitions are identical. However, when C varies with the voltage across it, care must be used in interpreting results since at any given value of voltage the charge on the capacitor will not necessarily equal the product of the capacitance and the voltage.

Conductance:

Symbol	XX  YY
Descriptor	SGXXYY
Characteristic	$i_b = Gv_b$ Branch current equation
Remarks	<p>The branch current equation shows that TAG defines conductance in the absolute sense rather than incrementally. When G is constant, the absolute and incremental definitions are identical. However, when G varies as a function of the voltage across it, care must be used in interpreting the results. G is equal to the slope of a line drawn from the origin of the i vs v curve to the operating point and does not relate directly to slope of this curve. For this reason, non-linear conductance is often best treated as a voltage dependent current source where the i_b vs v_b relationship may be used directly.</p>

Reciprocal Inductance:

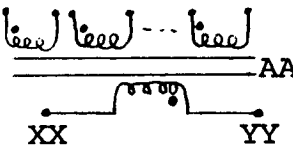
Symbol	XX  YY	Initial condition current is zero
Descriptor	SLXXYY	
Characteristic	$i_b = L \int_0^t v_b dt$	Branch current equation
Remarks	TAG uses the symbol L to denote reciprocal inductance rather than normal inductance. This is important to remember.	

All TAG inductors have zero initial condition current. Initial condition current must be modeled at the TAG equivalent circuit level by an ideal current source whose value is equal to the initial current. This current source is connected across the inductor and given a direction identical to the true initial current. The current source is treated as if it were actually part of the original circuit schematic.

The branch current equation shows that TAG defines inductance absolutely rather than incrementally. When L is a constant the absolute and incremental definitions are identical. However, when L varies with the integral of the voltage across it, care must be used in interpreting the results. L is equal to the slope of a line drawn from the origin of the i vs ϕ curve to the operating point and does not

relate directly to the slope of this curve. For this reason non-linear inductance is often best treated as a voltage integral dependent current source where the i_b vs ϕ_b relationship may be used directly.

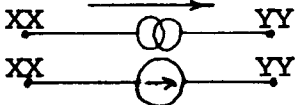
Ideal Transformer Winding:

Symbol		Transformer number AA
Descriptors	SNXXYY/AA-T	Relates this winding to Xformer No.AA, assigns to it T turns, and assigns the dotted terminal of all windings on AA to the second node position of the descriptor.
Characteristic	$\sum_{i=1}^n T_i \cdot i_{b_i} = 0$	The summation must include all windings of AA
	$\frac{V_{b_i}}{T_i} = K$	a constant for all windings of AA
Remarks	<p>Each winding requires its own descriptor. The dotted terminal of each winding of a given transformer must occupy the same node number position of all winding descriptors related to that transformer.</p> <p>The transformers must be numbered sequentially starting with number one.</p>	

The turns per winding must be expressed in integer form as the actual number of turns or as some conveniently normalized number of turns. Its most significant digit is placed in the space immediately following the dash. The range of integers allowed is 1 to 131072.

The characteristic transformer equations are applied by TAG to the circuit simulation equations as a secondary set of voltage and current constraints. This reduces the number of equations to be solved by eliminating all dependencies among the equation variables which result from the transformer constraints. The process is very similar to that of reflecting the impedances and sources seen by all windings into a single winding to consolidate the effects of all windings into a single equation.

Current Sources:

Symbol		Fixed current source Variable current source
Descriptor	SIXXY	Node order related to current flow

Characteristic	$ib = I$	Branch current equation
	$vb = ?$	Branch voltage is indeterminate

Remarks The relationship between current source direction, for positive values of source current, and the order of node identification numbers in the current source descriptor are shown above.

The current source is an extremely useful element for modeling non-standard components. This will be demonstrated further in the section covering non-standard elements. However, inductor initial conditions current is an example already discussed.

4. Connection List Format and Punctuation

The topological description of the circuit is entered into the TAG computational system as a connection list. The connection list contains an element descriptor for every branch in the TAG equivalent circuit plus certain punctuation described below. Since the computer accepts only punched cards as input, the user must prepare the connection list in such a way that it can easily be transferred to a set of punched cards by a key punch operator. IBM Fortran Coding Form (Form No. X28-73274) provides a convenient preset format for this operation. The example in the next section provides a picture of this form and a demonstration of its proper useage.

A standard IBM card contains 80 spaces, 72 of which are shown on the coding sheet. While connection list entries may be placed anywhere on the actual card, it is probably best to contain them within the 72 spaces provided on the coding sheet. The descriptor entries must be separated by a single comma but may be spaced at any distance desired, since blank positions on the card are ignored. The list of descriptors follows from card to card until all elements are properly included. As many cards as necessary may be used. The final entry must be followed by an asterisk to signal the end of the connection list. The separating comma and final asterisk are the only punctuation required or allowed within the connection list.

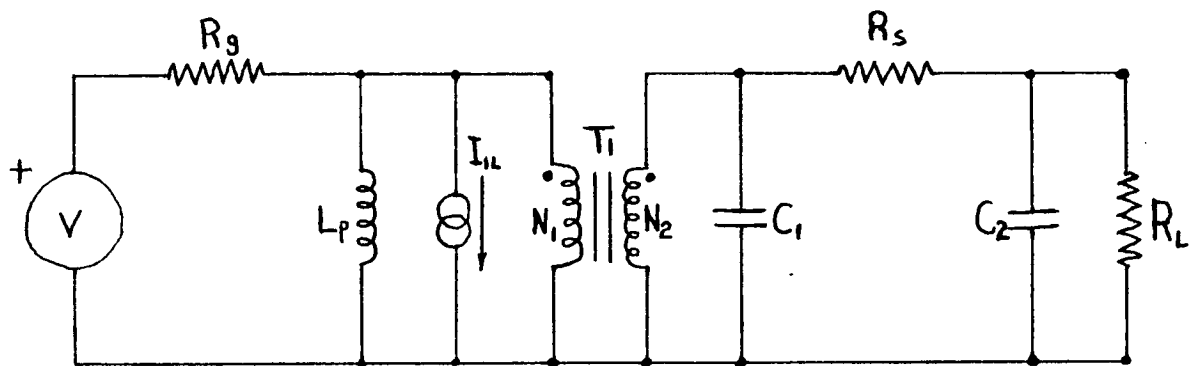
It must be emphasized here that only the descriptor entries, the separating commas, and final asterisk of the connection list are allowed within the topological description portion of the TAG description deck. The connection list does not even remotely relate to Fortran coding and can be placed upon any standard IBM card coding form. The Fortran coding form is suggested however, since it fits in well with the requirements of the rest of the TAG description deck.

The connection list should be ordered so that the elements having the largest numerical value come first. In many cases this will insure the optimization of the simulation equations in terms of accuracy of solution. However, this constraint may sometimes prove inconvenient because the component values to be tried may change magnitude from analysis to analysis or may simply not be known at the time that the connection

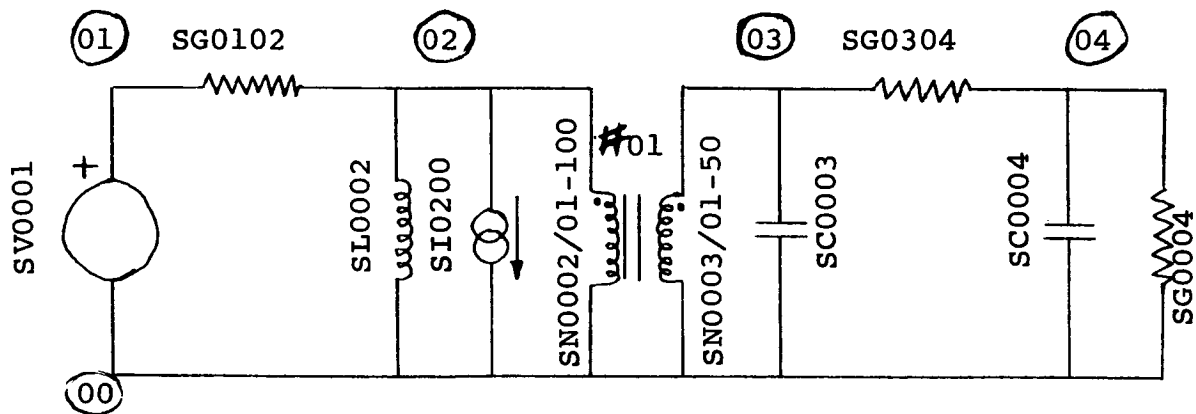
list is created. If the coding sheets are prepared with only one descriptor per line, the cards punched from these sheets will have no more than one entry each. The order may then be changed at will by a simple rearrangement of the cards. This added flexibility can also be seen to help accomodate simple changes in the circuit topology at a minimum effort.

5. Example Connection List Formulation

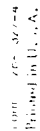
a. Initial Circuit Schematic



b. TAG Equivalent Circuit Schematic



c. Example Connection List Coding Sheet



Received 1977-05-24

3-20

C. Non-Standard Component Description

1. General Discussion

The TAG equation generator treats all components described in the connection list as standard TAG elements. A standard TAG element is defined by the fact that it belongs to the list of six basic two terminal elements recognized by the TAG processor and has a characterizing parameter which does not change value during any given simulation run. By so restricting the range of allowed component types treated by the equation generator, the simulation equations are simplified to a single standard formulation which is applicable to all types of circuits.

Treatment of components which can not be characterized as standard TAG elements must be accomplished by proper equivalent circuit modeling before the connection list is drawn up or by special treatment given to portions of the equations during the solution process. Equivalent circuit modeling handles devices whose structure is too complex to be adequately characterized by a single standard element. Special equation modification and solution techniques handle devices whose elementary parameters vary during simulation as a function of time or as a function of some circuit dependent variable. Real components are always non-standard to some extent since they all display some parasitic effects and non-linear behavior due to their finite size and imperfect internal mechanisms. TAG allows the user to model real devices to any degree of refinement required.

2. Multiple Element Devices

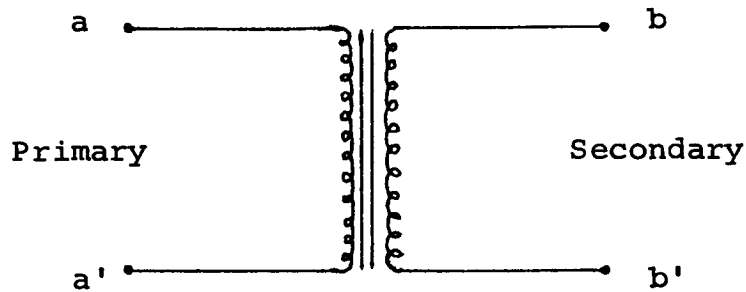
TAG provides a mathematical analog for six standard electrical elements capable of simulating the behavior of any lumped, linear, bilateral, finite network. It is therefore possible to model any electrical device no matter how complex as a subnetwork of standard TAG elements as long as its behavior adequately conforms to the laws governing such networks. The whole class of linear filter networks may be modeled to any degree of refinement desired by this very simple technique.

As discussed in the next several sections, TAG further provides a method of introducing non-linearities into the behavior of the standard elements as well as unilateral coupling between them. This allows one to model a much broader class of components including most active devices such as transistors, vacuum tubes, and magnetic amplifiers. The only constraints of this class are that the devices be representable by lumped, finite networks.

Equivalent circuit modeling of multiple-element devices must be accomplished while the TAG equivalent circuit is being constructed. Every element of a device subnetwork must appear in the TAG equivalent circuit to be processed by the equation generator. An equivalent circuit model for a real transformer is demonstrated below.

Transformer:

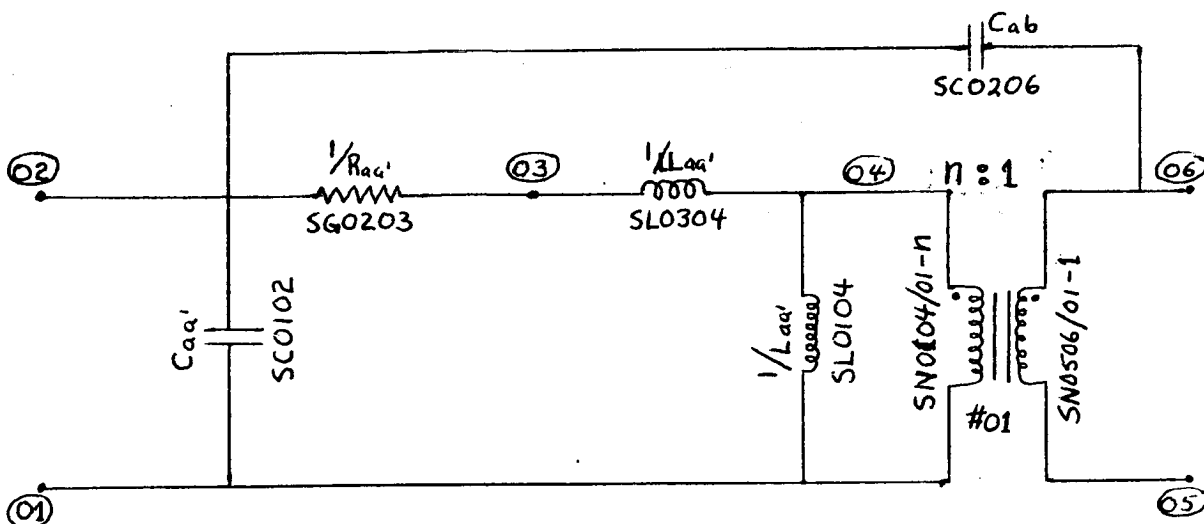
Schematic Representation:



Manufacturer's Specification:

<u>Parameter</u>	<u>Value</u>	<u>Conditions of Measurement</u>
Primary Inductance	$L_{aa'}$	measured at primary with secondary open
Turns Ratio	N	
Leakage Inductance	$L_{Laa'}$	measured at primary with secondary shorted
Primary Capacitance	$C_{aa'}$	measured at primary with secondary shorted
Primary to Secondary Capacitance	C_{ab}	measured with both windings shorted
Primary A.C. Resistance	$R_{aa'}$	measured at primary with secondary shorted

TAG Equivalent Network



The equivalent circuit shown is obviously built around the specification provided and only approximately models the actual device. However, if the only information available is that given in the specification, there is no point in trying to further complicate the model. The parameters of any more complicated a model would be impossible to evaluate from the given specifications. In fact, if the effect of any of the parasitic elements ($L_{aa'}$, C_{ab} , R_{aa} , $LL_{aa'}$) is of negligible significance, that element should be left out of the equivalent circuit.

3. Common Properties of Parameter Description Statements

a. General Characteristics

The characteristic parameter of any standard TAG element except the ideal transformer winding can be made to vary during a simulation run as a function of either the independent variable, time,

or certain dependent circuit variables. This is accomplished by means of a set of parameter description statements which are placed immediately following the connection list. These statements use the Fortran II computer language to describe function relationships between element parameters and circuit variables. These relationships are used to modify effected portions of the simulation equations between basic computational steps. The equations are updated to keep them in step with changes in both the time variable and the circuit state variables.

In order to use this feature effectively, it is absolutely essential that the user understand the Fortran II programming language. It should be stressed that the TAG description deck does not follow the rules for creating normal Fortran programs. However, it does borrow heavily from both the syntax and word structure of Fortran II. Several independent authors have written short text books covering Fortran programming; however, there are two documents available from the International Business Machine Corp. (IBM), which should sufficiently cover the subject for the average engineer. The first, "General Information Manual - FORTRAN", form No. F28-8074-1, is recommended as a first reader on Fortran programming. The second, "IBM Systems Reference Library - IBM-7090/7094 Programming Systems - FORTRAN II Programming", form No. C28-6054-5, is recommended as a reference manual for engineers who have already had some exposure to computer operations and

Fortran programming. For the remainder of this section, it will be assumed that the reader has some familiarity with the rules for structuring Fortran variable names, arithmetic statements, functions, and subroutines. However, some commentary and many examples will be provided so that a relative novice can perform some level of meaningful analysis by merely following the examples given.

b. An abbreviated List of Basic Fortran Rules

A few of the basic rules of Fortran will be listed here to provide a basic reference and eliminate the necessity of repeating them later in the text. Either of the two manuals referenced above should provide adequate further clarification.

- 1) Each column on a Fortran coding sheet corresponds to a character position on an IBM punched card.
- 2) The first five columns are used to number Fortran statements for cross referencing purposes. Only statements which are referred to by other statements in the program need be numbered. The numbers can normally range from 1 to 32,768, however the TAG generated solution program uses some numbers greater than 5999.
- 3) Column six is reserved for continuation. The cards of a multiple card statement must be numbered sequentially from 0 to 9 in column 6. No greater than 10 cards are allowed for a single statement.

- 4) Columns seven through seventy-two are used to express the statement itself. Blanks may be used freely in order to improve the readability of the statement and are ignored by Fortran.
- 5) Columns seventy-three through eighty are not used by Fortran and may therefore be punched with anything. Column seventy-three is used by TAG as will be explained in section F.
- 6) Variable names may be of no greater length than six characters and must start with an alphabetic character.
- 7) Variable names beginning with the letters I, J, K, L, M or N are called fixed point variables and may have integer values only.
- 8) All other variables are floating point variables and may have values whose magnitudes lie between 10^{38} and 10^{-38} .
- 9) Any single variable name may refer to a list of values. Any single member of the list is referred to by subscripting the list name with the numerical position of the desired value within the list. Thus V(10) refers to the tenth value in the list V.
- 10) The arithmetic operations allowed by Fortran are listed below with their symbolic operators. The list is given in the order in which the operations are performed within a given expression.

**Exponentiation

* and / Multiplication and Division

+ and - Addition and Subtraction

- 11) Floating point and fixed point variables may not be used within the same arithmetic expression (Exponentiation is an exception).
- 12) Parentheses should be used to clarify the order of operations within an arithmetic expression. The operations will be executed starting with the inner most parentheses and working toward the outermost. Parentheses must always occur in pairs.

c. Useage of Fortran Statements

- 1) Discussion and Use of Simple Arithmetic Statements

Example: $SV0001 = RATE * FT$

The two types of Fortran statements used to describe component parameter dependencies are the arithmetic statement and the CALL (Subroutine) statement. The arithmetic statement is used to describe relatively simple parameter dependencies which can be expressed as single self-contained mathematical expressions. The only variable that may appear to the left of the equal sign in such an expression is the name of the characterizing parameter of a specific TAG element which appears in the connection list. This allows TAG to properly classify the statement as a parameter description statement. To the right of the equal sign may appear any desired arithmetic expression made up of properly constructed

Fortran variables and functions correctly connected by allowed Fortran operations symbols.

Each arithmetic statement must be self-contained in the sense that it requires no other parameter description statements to supply it with precalculated parameter or variable values. Complicated expressions requiring up to ten lines of Fortran coding may be accommodated by using the statement continuation feature of Fortran.

Arithmetic expressions are restricted to describing the dependency of a single parameter and must therefore be restated in full for every element parameter which behaves in a similar manner. For simple or seldom used expressions, this is no great handicap. However, there are many cases where a single set of complex arithmetic expressions may be used not only many times in the same circuit description, but for many different circuits. An example of this might be the mathematical model of a transistor. In this case, it would be extremely useful to be able to create a set of expressions which are independent of any specific circuit and can be incorporated into the TAG system in such a way as to be available to the user whenever desired. The Fortran subroutine and function capability provide an ideal framework for handling such situations.

2) Description and Use of Fortran Functions

Example: $SI0001 = AMPS * (1. - EXPF(-FT/TAU))$

For a definitive discussion of the properties and proper utilization of the subroutine and function capabilities of Fortran, the reader is again referred to the various available Fortran manuals and texts. However, a few remarks will be made here covering the usage of these two devices within the TAG system.

The Fortran function provides a means of incorporating a complicated single valued mathematical expression into an arithmetic statement as a single symbolic name. The function name as it appears in an arithmetic statement is used to mean the value of the function. The arguments of the function which appear in parentheses following the function name are the independent variables and parameters required to calculate the value of the function. The values of the arguments must be either pre-calculated or supplied to the program as input data. A function is not a self-sufficient arithmetic statement but usually occurs as part of the right hand side of a Fortran arithmetic statement.

Using the rules outlined in the Fortran manual, the user may construct any function he desires. However, as a convenience, many often required mathematical functions are already built into the Fortran system and

can be called on by simply referring to them by name and supplying them with proper argument names and values. An abbreviated list of these functions is provided below. A complete list of system functions and their useage can be obtained from the Computer Systems Group.

Function Type	Definition	Fortran Name
Absolute Value	$ X $	ABSF(X)
Exponential	e^X	EXPF(X)
Natural Logarithum	$\log e^X$	LØGF(X)
Trigonometric Sine	$\sin(X)$	SINF(X)
Trigonometric Cosine	$\cos(X)$	CØSF(X)
Square Root	\sqrt{X}	SQRTF(X)

Abbreviated Table of System Provided Functions

3) Description and Use of Fortran Subroutines

Example: CALL DIØDE (SV0609, \$SI0906,\$SC0609,
DATA)

The Fortran subroutine provides a second more powerful means for expressing functional relationships between variables which are independent of any specific program. Fortran subroutines are called into use by means of a CALL statement which specifies the name of the subroutine under which the operations performed by the subroutine are defined. In addition, the CALL statement contains a list of arguments which defines all dependent, independent, and parametric variables required to perform the prescribed operations and define the results.

Unlike the Fortran function, a subroutine CALL statement stands alone as a parameter description statement, and may calculate the values of as many dependent variables as named in the argument list.

A single subroutine may be used to describe the whole set of model parameter interdependencies which occur in complex devices. Such a subroutine which is independent of any specific device may be called upon to describe all devices of the same general type. The TAG parameter description list will contain a single subroutine CALL statement for each such device appearing in the TAG equivalent circuit. The subroutine itself, which defines the relationships between the device variables, is not included in the TAG description deck but must be placed in the TAG system deck between the description and binary execution decks. In this way it automatically becomes a part of the system.

At least one of the arguments of a subroutine call statement must be the name of the characterizing parameter of a specific branch element which appears in the connection list. In addition, this parameter must be a dependent variable of the subroutine and, therefore, must be calculated by it. In order for TAG to recognize subroutine CALL statements as dependent parameter statements, at least one of the dependent variable names appearing in

the argument list must be prefixed by a dollar sign (\$) character. To guarantee proper statement classification by TAG, it is recommended that all dependent parameter variables which appear in the argument list of a subroutine CALL statement be prefixed by a dollar sign (\$). For example the dependent variable SI0103 should be listed \$SI0103 when it appears in a CALL statement.

As in the case of the Fortran function the user may prepare subroutines which suit his own special needs, or, when applicable, he may use subroutines previously added to the system. These system subroutines will not be listed at this point in the discussion since their useages are not very general. However, some of them will be presented later when examples of their application arise.

d. Naming Variables for Parameter Description Statements

1) Dependent Variables

Example: SV0102, SI0304, SG0914

The most important class of dependent variables appearing in parameter description statements are the element parameters whose values are calculated by the statements. In fact, in arithmetic statements this is the only type of dependent variable allowed. Each of the element parameters appearing in a parameter description statement must belong to a single specific branch element of the connection list.

TAG takes advantage of this fact by allowing the descriptor used to identify a particular element in the connection list to be used again to identify the characterizing parameter of that element in the parameter description list. This reduces considerably the number of different variable names that the user must manufacture and keep track of. The only modification to this rule occurs when the parameter descriptor appears as an argument of a subroutine CALL statement. In that case, it is prefixed by a dollar sign (\$) character.

2) Function Parameters and Relative Constants

Example: AN, AI, TI, CID1

Function parameters and relative constants must be assigned names which follow the rules of Fortran II language. In general, this means that such names will begin with an alphabetic character and be no more than six characters in length. The Fortran distinction between the useage of names beginning with the letters I, J, K, L, M, & N and all others must be strictly adhered to. In addition to these Fortran restrictions it is recommended that names beginning with the letters S, F, & L not be used within the TAG description deck as they might possibly coincide with names already used by the TAG system.

3) Independent Variables

Example: FT = Simulated time
SV0104 = Voltage at node 04 relative
to that at node 01
SS0104 = Integral of Voltage SV0104

The three variables named and defined in the above examples are the only types of independent variable allowed within the parameter description list. Except for appropriate changes in node numbering, these three variables must appear with their applicable parameter description statements exactly as shown. The use of these three variables will be discussed and demonstrated in detail in the next section.

Note that the symbol used to designate node pair voltage integral follows the standard format established for TAG element descriptors. Its distinguishing feature is the S appearing in its second character position. The polarity rule for this descriptor is the same as that for the node pair voltage whose integral it represents.

4) TAG Control Variables

Example: LALGFT = 1 on first pass through
computations
= 2 on all other passes

LLCNT = -1 on all normal passes
through the solution program
= N, a dependent stop function
identification number, when-
ever a dependent stop function
is satisfied

The control parameter LALGFT is provided by the TAG system to allow initializing of parameter description subroutines. Its value is set to 1 by TAG during the first pass through the simulation calculations. Thereafter, it is set to 2. When used in a subroutine it must appear in the appropriate position in the argument list of the call statement.

The control parameter LLCNT is provided by TAG to flag the occurrence of a dependent variable stop. Each stop function variable is assigned an identification number N. At the exact point at which the stop function is satisfied (i.e., the stop function variable equals zero) LLCNT is set equal to N for one evaluation of all dependant stop functions. At all other times LLCNT equals -1.

Other control variables provided by TAG are generally of no great value to the user and will not be covered here. In general these other variables are of significance only within the inner computational sequence of the TAG generated solution program.

4. Time Dependent, Parameter Description Statements

a. General Characteristics

Any parameter description statements whose independent variable is time is classified as a time dependent statement. Such statements may be

created to describe the behavior of the characterizing parameter of any of the standard TAG elements which appear in the connection list except ideal transformer windings.

Time varying, parameter description statements are obviously applicable to transient analysis only. Their basic effect is to transform a circuit which the equation generator assumes is a linear time invariant network into one in which element values change as time changes. The effect of this transformation is to force some of the coefficients and forcing functions of the simulation equations to vary in value as functions of simulated time. Time dependencies have no effect upon the linearity of the circuit or the simulation equations which characterize that circuit.

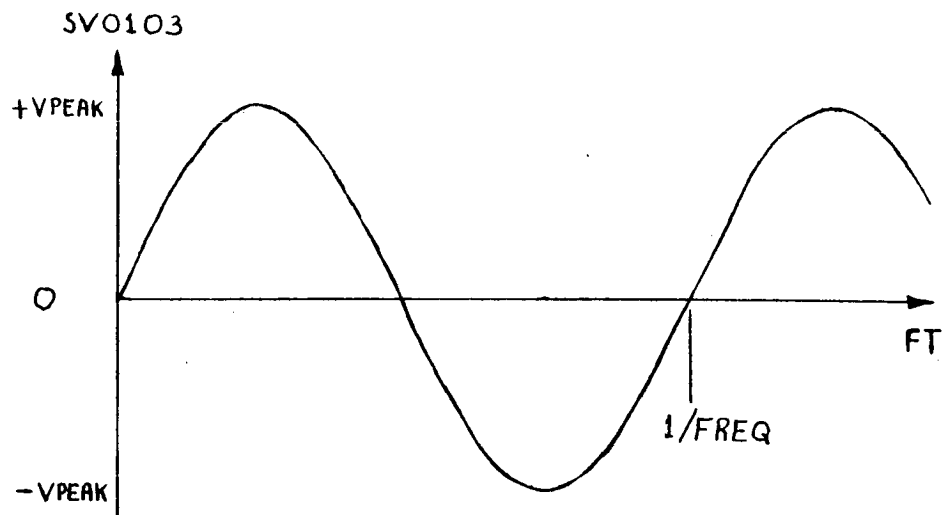
Time dependent statements are most often used in the description of voltage and current sources which appear in circuits as driving sources. The flexibility provided by TAG in the formulation of driving source functions is limited only by the ability of Fortran II to describe such functions. Both continuous and discontinuous time functions can be accommodated by TAG but require somewhat different treatment.

b. Continuous Time Functions

Examples: $SV0103 = VPEAK * \text{SINF} \quad (6.28 * \text{FREQ} * \text{FT})$
 $SI0006 = \text{AFINAL} * \quad (1. - \text{EXPF}[-\text{FT}/\text{TAU}])$

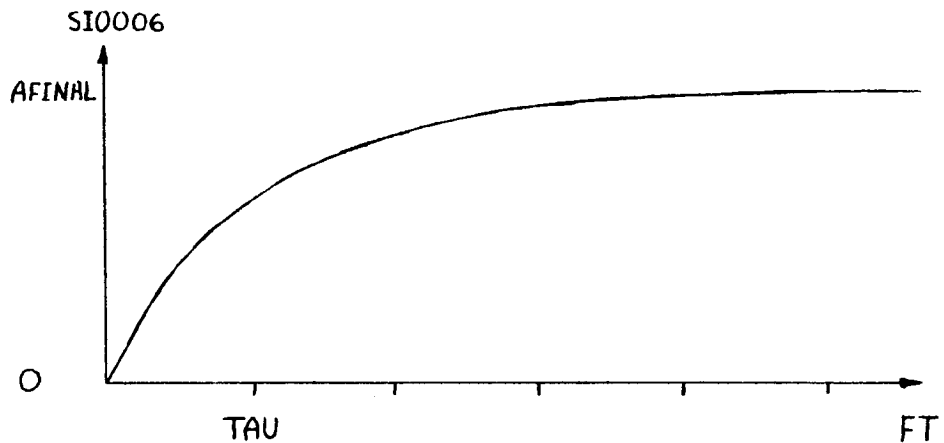
Characterizing parameters whose values change as a continuous function of time may be described by the simplest arithmetic statement or subroutine that will produce the desired functional relationship. Two such statements are shown above as typical examples.

The first describes a sine wave voltage source whose peak value is equal to the value of VPEAK and whose frequency is equal to the value of FREQ. When the value of the right hand side of the equation is positive the voltage at node 03 is positive with respect to the voltage at node 01. This convention is entirely consistent with the rules for forming element descriptors. The numerical values of VPEAK and FREQ are entered as part of the input data list. This waveform is pictured below.



The second example describes a current source element whose current is exponentially rising to a final value equal to AFINAL at a time constant

whose value is equal to TAU. When the value of the right hand side of the equation is positive the current will flow within the source element from node 00 to node 06. The values of AFINAL and TAU are entered as part of the input data list. This wave form is illustrated below.



c. Discontinuous Functions of Time

1) Discontinuities at Time $FT = 0$

Examples: $SV0003 = VIN$

Normal power supply turn on transient

At the beginning of transient solution runs TAG does not automatically initialize capacitor voltage or inductor currents to a set of values which are consistent with the D.C. steady state implied by the initial values of voltage and current sources. The initial state from which the transient solution will proceed must always be precomputed by the user and entered into

the data list as initial capacitor voltage and inductor current values. Careful "hand" techniques or the D.C. steady state analysis capability of TAG may be used to compute the required values.

Any circuit state variable which is not specifically initialized by the user will be given an initial value of zero. Where no initial conditions are specified, the circuit will start out in the ground state, defined as the zero energy state. Transient analyses which proceed from the ground state produce a simulation of the idealized power supply turn on transient in which all voltage and current sources rise to their initial values in zero time.

This method of producing power supply turn on transient analysis can be generalized to produce any desired type of step function analysis. By properly initializing the circuit state variables to one set of energy source conditions, the step function response to any energy source can be produced by simply entering a value for that source in the data list which is different from the value for which the given initial conditions are valid.

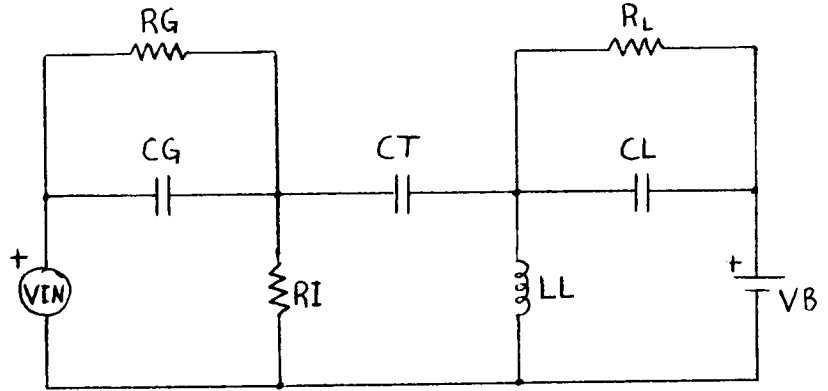
This can also be accomplished less directly by creating a parameter description statement like that shown in the first example above. In the data list, VIN is assigned the desired initial

starting value and SV0003 is assigned the value which is consistent with the initial circuit state.

Some caution must be exercised in applying voltage and current source steps to arbitrary networks. The concept that the energy state of a storage element cannot change in zero time applies only to real circuit components in which there are always parasitics present which maintain a finite bound on values of current and voltage. In certain topological configurations the idealized TAG capacitor or inductor may be required to change energy states instantaneously by the application of a step of voltage or current to the circuit. TAG is not set up to handle such situations and will generally produce wrong answers when they occur. For this reason circuit topologies which produce such conditions must be avoided. The two general cases in which they occur will be discussed below.

The Voltage Driven Capacitor Loop:

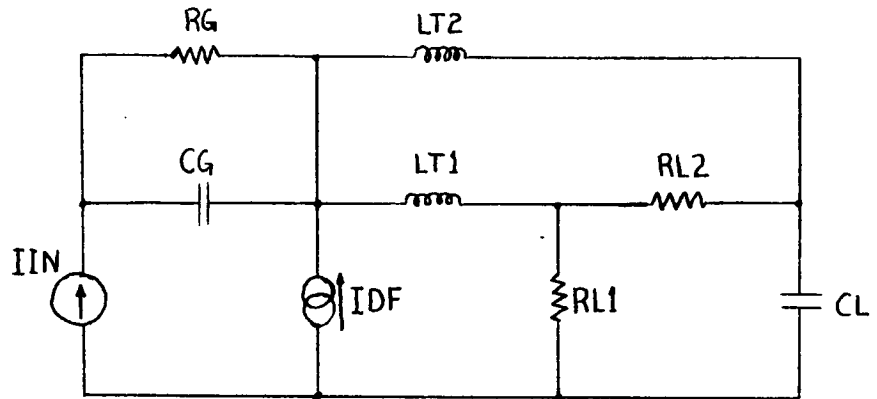
The voltage driven capacitor loop occurs in a network whenever it is possible to trace a closed path of branch elements in which every element in the path is either a voltage source or a capacitor. This is illustrated schematically below where the elements V_{in} , C_g , C_T , C_L , and V_B form such a loop.



It is obvious that if V_{in} applies a step of input voltage to this circuit the charges in the capacitors must change in zero time creating a loop current of infinite amplitude. TAG will assume that the whole voltage step appears across only one of the capacitors (usually the smallest). This problem must be circumvented either by removing the smallest non-essential capacitor from the loop or by inserting a series resistor or inductor into the loop of the largest value that will not adversely distort the simulation.

The Current Driven, Inductor Cut-Set:

The current driven, inductor cut-set occurs whenever it is not possible to find at least one path in a network that connects the two nodes in which none of the branch elements are either current sources or inductors. This is illustrated schematically below where the elements I_{in} , I_{A+} , L_{T1} and L_{T2} form a cut-set of nothing but current sources and inductors.



It is obvious that if I_{in} applies a step of input current to this circuit the flux stored in inductors L_{T1} and L_{T2} must change in zero time creating an infinite voltage across the cut-set. The node voltage at the top of the current source, I_{in} , can not be calculated at the time of the current discontinuity. If this voltage is necessary it can be made available only by shunting one of the current sources by either a resistor or capacitor.

d. Discontinuities at times other than $FT=0$

Example:

```
SI0203 = ASTEP*UTF(TS)
SV0102 = VPEAK*(1.-UTF(T1))
SG0901 = GMAX*(-UTF(TA)+UTF(TB))
SV0010 = FT*VP/(TR+TD)*(-UTF(TD)+UTF(TR))+VP*
1(-UTF(TR)-UTF(TP))-FT*VP/(TF-TP)*(-UTF(TP)+UTF(TF))
```

At simulated time, FT , other than zero, time function discontinuities must be specified by means of the special switching function, UTF , provided as part of the TAG system. The UTF function provides the solution program control operations required to accommodate abrupt changes in the values of simulation equation coefficients and forcing functions. In addition, $UTF(T)$ is itself available as a function whose value changes from one to zero at the exact time that FT becomes equal to T . Finally, whenever a UTF function changes state, the output and control sequence (described in section F) is executed (once with the value of $UTF = 1$ and again with the value of $UTF=0$).

The UTF function is defined as follows:

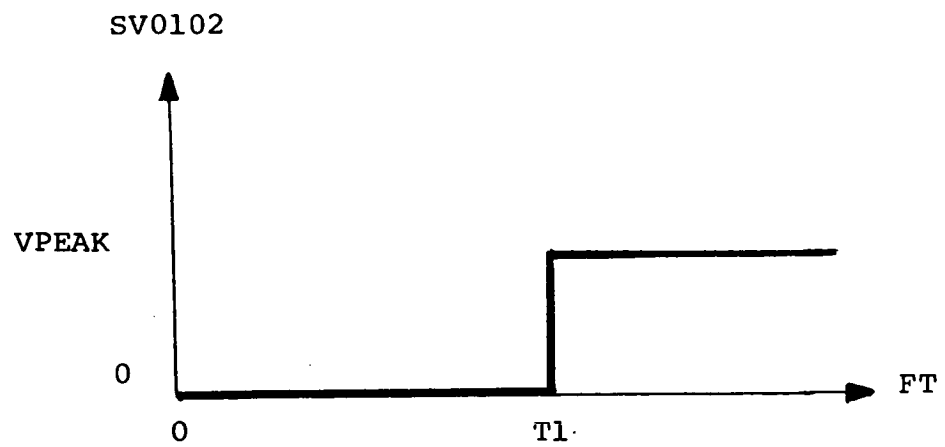
$$\begin{aligned} UTF(X) &= 1 && \text{for } FT \leq X \\ &= 0 && \text{for } FT > X \end{aligned}$$

The following waveforms refer to the examples shown above.

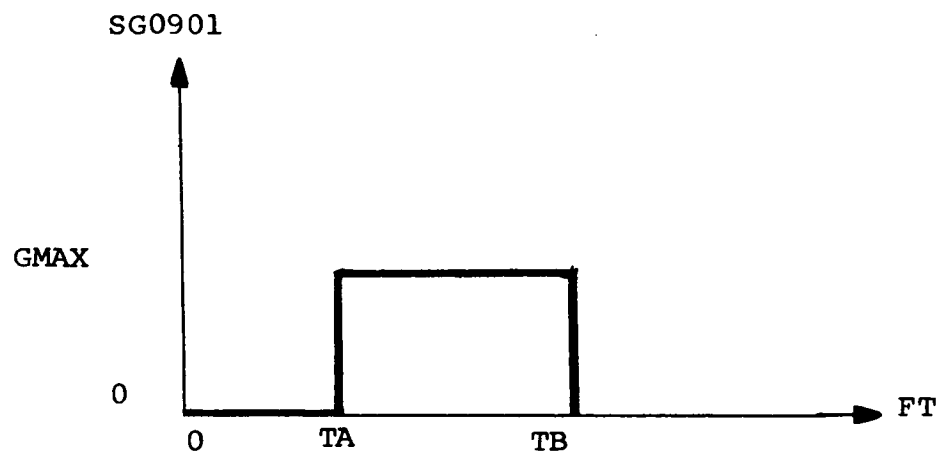
Example 1



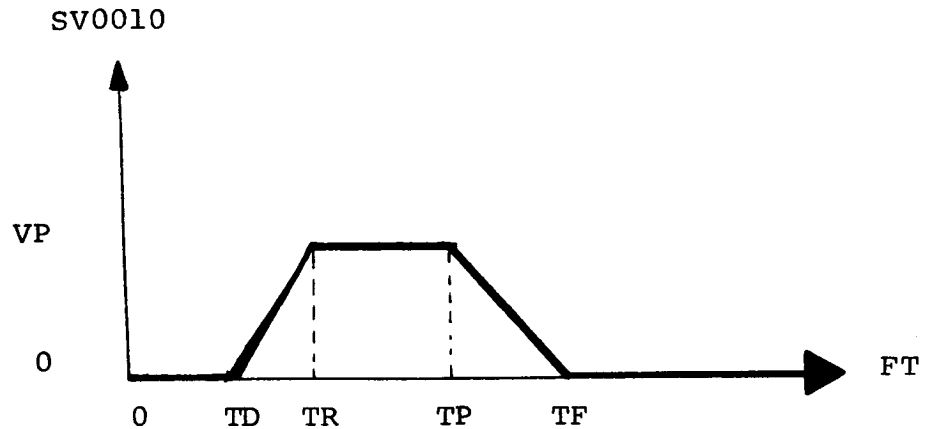
Example 2



Example 3



Example 4



From the above display of examples, the ability of the UTF function to provide arbitrary discontinuous time functions is well demonstrated. However, the function segments between UTF discontinuities need not be linear or constant as shown. The UTF function may be treated like any other general Fortran function appearing in the parameter description statements.

5. Circuit Dependent, Parameter Description Statements

a. General Characteristics

Any parameter description statement expressed as a function of a circuit dependent variable is classified as a circuit dependent statement. The range of independent variables allowed for such statements is restricted by TAG to node pair voltage and node pair voltage integral. Circuit dependent statements may be created to describe the behavior of any of the standard TAG elements except voltage sources and ideal transformer windings.

The circuit dependent statement creates within TAG the ability to cope with non-linear element behavior and non-bilateral interbranch coupling. Since these two properties represent general attributes of many active components, modeling of most complex active devices is made possible by this capability. This obviously greatly enhances the overall usefulness of TAG.

Circuit dependent, parameter description statements are applicable to both D.C. steady state and transient analysis. Their usual effect is to transform the standard simulation equations generated by TAG into non-linear, non-constant-coefficient equations. In the transient case this only slightly complicates the solution process over that required for time varying circuits. However, circuits which

require the use of circuit dependent statements will often create more complicated equations than those without and may, therefore, take longer to solve. Impedences which greatly effect local time constant behavior may be masked in such circuits by the particular method used to describe element dependencies. Local time constant behavior may greatly effect the number of computational steps required to perform a transient simulation and, therefore, greatly effect the solution time.

In the DC steady state case the solution process is greatly complicated by the introduction of circuit dependent statements. The sets of non-linear algebraic equations thus generated are in general difficult to solve because there is no explicit form in which to express their solution in terms of equation coefficients and forcing functions. The process used by TAG to generate solutions to such equations is called the Newton-Raphson method. It uses an iterative algorithm for continually improving the correctness of some initial solution estimate. The initial estimate is provided either by the program or the user and must be within a certain distance of the actual solution in order to insure process convergence. Such initial solution estimates are at present difficult to enter into TAG without extra programming at the solution program level.

TAG automatically assumes the ground state to be the initial state for all D.C. steady state simulation runs. It will therefore generally achieve convergent solutions when all energy source values are sufficiently close to zero. By stepping the values of all energy source elements from zero to their final values in sufficiently small increments, it is usually possible to achieve D.C. steady state solutions to any arbitrary distribution of energy source values without resorting to extra coding at the solution program level.

Circuit dependent statements are most often used for modeling non-standard electronics components. The flexibility provided by TAG for formulating complicated functional dependencies is limited only by the ability of Fortran II to describe such functions. For D.C. steady state problems a circuit parameter dependency must be expressed in terms of its independent variables as a continuous function having a reasonably well behaved derivative defined over its entire range.

For transient problems circuit parameter dependencies must be either smoothly continuous or piecewise continuous over their entire range of use. Smoothly continuous functions are the most easily handled and do not vary in form from the functions required by the D.C. steady state process.

Piecewise continuous functions are only allowed when restricted to a finite number of pieces over their range of use. Every boundry between two adjacent pieces of such a function must be flagged by the use of a dependent stop function defined at that boundry. The dependent stop function provides the solution program control required to handle abrupt changes in the values of the coefficients and dependent driving sources of the simulation equations. Such a stop function is in no way similar to a normal Fortran function. Its formulation is dependent entirely upon the subroutine F mark which is used in the solution program to solve transient simulation equations.

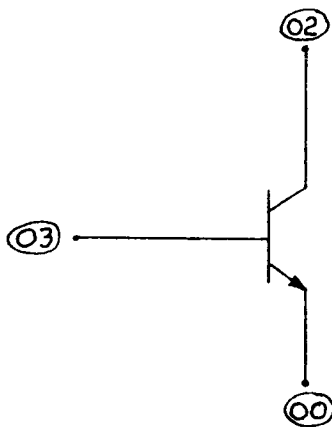
b. Smoothly Continuous Circuit Dependent Functions

Examples: $SI0201 = AN * SV0401 * SG0401$
 $SI0100 = IS * (EXP(SV0001/VO) - 1.)$
 $SI0201 = AN * IS * (EXP(SV0001/VO) - 1.)$
 $CALL\ TRAN3(SV0001,SV0201,\$SI0100,\$SI0102,$
 $\$SC0001,\$SC0201,DATAT,CIE,CIC,LALGFT,$
 $KTDC)$

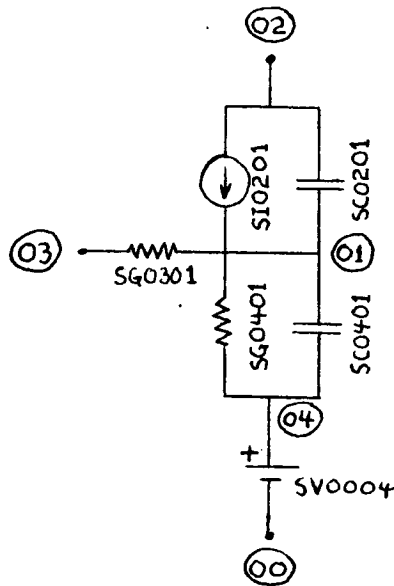
Element parameters whose values change as smoothly continuous functions of node pair voltage or node pair voltage integral may be described by the simplest arithmetic statements or subroutines that will produce the desired functional relationships. As these statements increase in complexity due to sophistication or generalization of a device model, a point is reached at which it becomes more efficient

to describe the device by a general subroutine which can model all devices of the given class. The process of increasing model sophistication is demonstrated in the above example of three different transistor models.

The first example demonstrates the circuit dependent statement required to describe the simple transistor model shown below. Here the current generator, SI0201, represents a collector junction collecting current at a rate of AN times the current in the linearized emitter junction, SV0004 and SG0401. The dynamic model shown can be reduced to a static model by eliminating the junction capacitors. The voltage source in series with the emitter conductance represents the emitter junction intercept voltage. The model shown is for an NPN transistor; however, by redefining polarities it could easily be converted into a PNP.



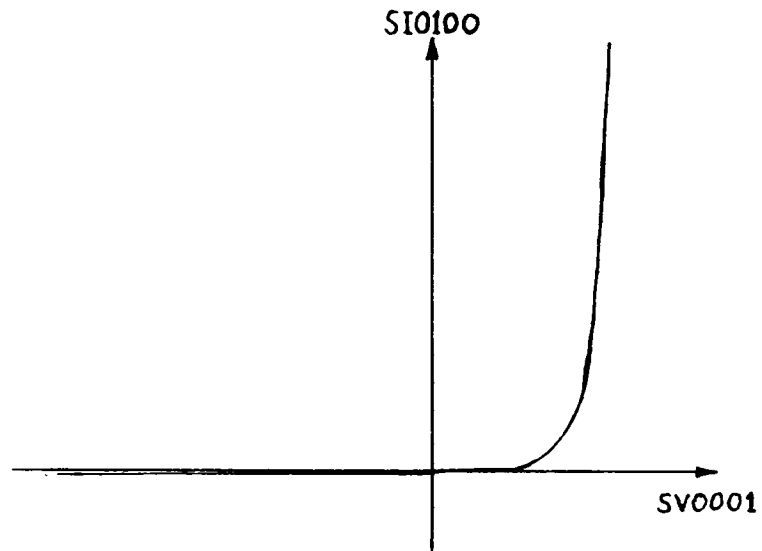
Transistor Schematic Symbol



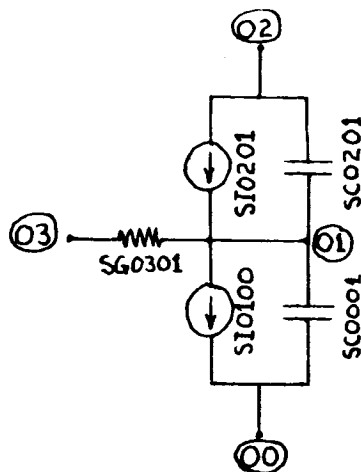
Simple Linear Transistor Equivalent Circuit

The second example statement is immediately recognized as the classical non-linear diode model in which I_S represents the diode saturation current and V_0 the equivalent thermal potential KT/Q . However, within the context of the present development, this statement will be used to describe the emitter junction of the simplified Ebers and Moll transistor model shown below. The combination of the second and third statements are sufficient to model the D.C. emitter and collector behavior for this device. The model gives reasonably accurate results in the forward active region for a large class of devices for which the inverted common

base current gain A_I is very small (less than .1). The junction capacitances shown in the model are assumed to be linear over the range of operations. The non-linear junction capacitance relationships could be added; however, they are complicated enough to warrant the creation of a general subroutine for the entire transistor model.



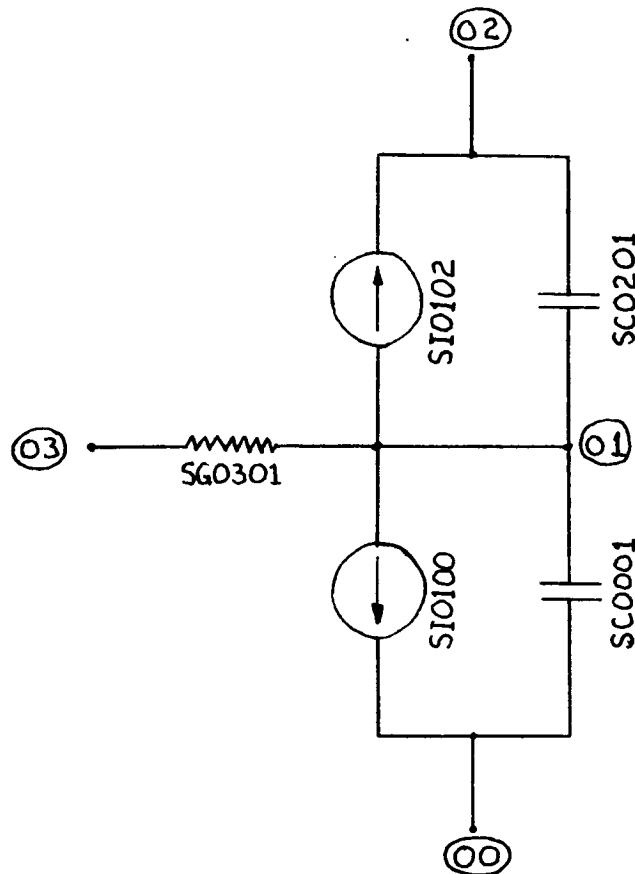
Classical Exponential Diode Curve



Simplified Ebers and Moll Transistor Equivalent Circuit
3-53

The generalized Ebers and Moll transistor model is complicated and useful enough to be coded into a subroutine which is independent of any specific transistor or circuit topology. TRAN3 is the name of just such a subroutine. The Fortran coding for this subroutine is reproduced immediately following the equivalent circuit illustration. The comment statements which appear as a part of this coding provide an adequate description of the subroutine arguments and model computations. This subroutine, when key punched onto IBM cards, forms a card deck which is placed between the TAG description deck and TAG binary execution deck during simulation runs. The Fortran system will compile and punch onto cards a binary version of the subroutine which may be added to the TAG system as a part of the binary execution deck.

Once the TRAN3 subroutine is a part of the TAG system, it may be called into use in the parameter description list by a CALL statement of the type shown in the fourth example. One CALL statement is required for each transistor model appearing in the TAG equivalent circuit. The parameter values of particular circuit elements required in the CALL statement argument list are named using the same descriptor vocabulary developed for the connection list. All other arguments must either be model parameters which are entered as constants in the input list or TAG control variables.



Generalized Ebers and Moll Transistor Equivalent Circuit

Ebers and Moll Transistor Model Subroutine:

Subroutine TRAN3(V1,V2,FI1,FI2,CCE,CCC,DATA,FI1I,FI2I,
LALGFT,KTDC)

C Ebers and Moll Non-Linear Transistor Model

```

SUBROUTINE TRAN3 (V1,V2,FI1,FI2,CCE,CCC,DATA,FI1I,FI2I,LALGFT,KTDC1
EBERS AND MOLL NON-LINEAR TRANSISTOR MODEL
C ARG(1) = V1          EMITTER DIODE VOLTAGE (PLUS FOR P POSITIVE)
C ARG(2) = V2          COLLECTOR DIODE VOLTAGE (PLUS FOR P POSITIVE)
C ARG(3) = FI1         EMITTER BRANCH CURRENT (PLUS FOR FLOW P TO N)
C ARG(4) = FI2         COLLECTOR BRANCH CURRENT (PLUS FOR FLOW P TO N)
C ARG(5) = CCE         TOTAL EMITTER SHUNT CAPACITANCE
C ARG(6) = CCC         TOTAL COLLECTOR SHUNT CAPACITANCE
C ARG(7) = DATA       TRANSISTOR PARAMETER ARRAY
C ARG(8) = FI1I        INITIAL VALUE OF EMITTER CURRENT
C ARG(9) = FI2I        INITIAL VALUE OF COLLECTOR CURRENT
C ARG(10)= LALGFT      FLAG = 1 ON FIRST PASS THROUGH SUBROUTINE
C ARG(11)= KTDC        FLAG SET TO 1 FOR DC CASE , TO 0 FOR TRANSIENT
C BULK RESISTANCES MUST BE INCLUDED IN EXTERNAL CIRCUIT IF DESIRED

```

```

C      DATA(1) = ISE      DATA(2) = ISC      DIODE SATURATION CURRENT
C      DATA(3) = GLE      DATA(4) = GLC      DIODE LEAKAGE CONDUCTANCE
C      DATA(5) = AN       DATA(6) = AI       COMMON BASE CURRENT GAIN
C      DATA(7) = TN       DATA(8) = TI       RECOVERY TIME CONSTANT
C      DATA(9) = VKE      DATA(10)= VKC      DIODE CONTACT POTENTIAL
C      DATA(11)= NE       DATA(12)= NC      DIODE GRADING CONSTANT
C      DATA(13)= KE       DATA(14)= KC      DIODE CAPACITANCE CONSTANT
C      DATA(15)= CES      DATA(16)= CCS      DIODE STRAY CAPACITANCE
C      DATA(17)= VO       THERMAL POTENTIAL KT/Q
      DIMENSION DATA(17)
      IF(KTCD-1) 10,1,1
1  IF(LALGFT-1) 5,5,10
C      INITIALIZE DIODE CURRENTS
5  FI1 = FI1I
   FI2 = FI2I
   GO TO 15
C      EMITTER AND COLLECTOR CURRENT CALCULATIONS
C      FIE= ISE*(EXPF(V1/VO)-1)/(1-AN*AI)
C      FIC= ISC*(EXPF(V2/VO)-1)/(1-AN*AI)
C      FI1= FIE - AI*FIC + V1*GLE
C      FI2= FIC - AN*FIE + V2*GLC
10 AN = DATA(5)
   AI = DATA(6)
   D = 1.-AN*AI
   SIE= DATA(1)/D
   SIC= DATA(2)/D
   FIE= SIE*(EXPF(V1/DATA(17))-1.)
   FIC= SIC*(EXPF(V2/DATA(17))-1.)
   FI1= FIE-AI*FIC + V1*DATA(3)
   FI2= FIC-AN*FIE + V2*DATA(4)
   IF(KTDC-1) 11,15,15
C      EMITTER AND COLLECTOR SHUNT CAPACITANCE CALCULATIONS
C      CJE= KE/(VKE-V1)**NE      EMITTER JUNCTION DEPLETION CAPACITANCE
C      CJC= KC/(VKC-V2)**NC      COLLECTOR JUNCTION DEPLETION CAPACITANCE
C      CDE= (FIE+SIE)*TN/VO      EMITTER DIFFUSION CAPACITANCE
C      CDC= (FIC+SIC)*TI/VO      COLLECTOR DIFFUSION CAPACITANCE
11 DE = DATA(9) -V1
   DC = DATA(10)-V2
   CJE= DATA(13)/DE**DATA(11)
   CJC= DATA(14)/DC**DATA(12)
   CDE= (FIE+SIE)*DATA(7)/DATA(17)
   CDC= (FIC+SIC)*DATA(8)/DATA(17)
   CCE= CJE+CDE+DATA(15)
   CCC= CJC+CDC+DATA(16)
15 CONTINUE
   RETURN
   END

```

c. Piecewise Continuous, Circuit Dependent Functions

Examples: $\$1STOP = VSTOP - SV0004$

```
CALL PLIND(SS0102,RIND,STATE,CISAT,  
           $1FLXST1, $2FLXST2,FLXIN1,DATA1,  
           LALGFT1)  
SIO201 = RIND*SS0102-STATE*CISAT
```

Element parameters whose values change abruptly at specific values of a node pair voltage or voltage integral must be described by the use of dependent stop functions in the parameter description list. The dependent stop function allows discontinuities to occur in the transient simulation equations at the exact circuit state for which the stop function has a zero value. Under no other conditions should parameter value discontinuities be allowed to occur except when a UTF function changes state.

Whenever a dependent stop function goes to zero the following operations are performed by the TAG solution program. First the differential equation integrator, a subroutine called F mark, stops integrating at the exact point in simulated time, FT, for which the stop function is, for all practical purposes, equal to zero. Next the TAG control variable, LCNT, is set to equal three plus the value of the stop function identification number. The program is then routed to the output and control sequence, described in detail in section E. Statements may be included in this sequence which change the values of circuit element parameters specifically when LCNT is equal to some number greater than three. A second TAG control variable, LLCNT, which is normally equal to -1 is set equal to the stop function identification number during the next evaluation of all stop function values.

integration subroutine is restarted as if a brand new simulation run was beginning using the final values of the previous simulation for its initial values. This sequence of events must accompany any abrupt changes in the coefficients or forcing functions of the simulation equations.

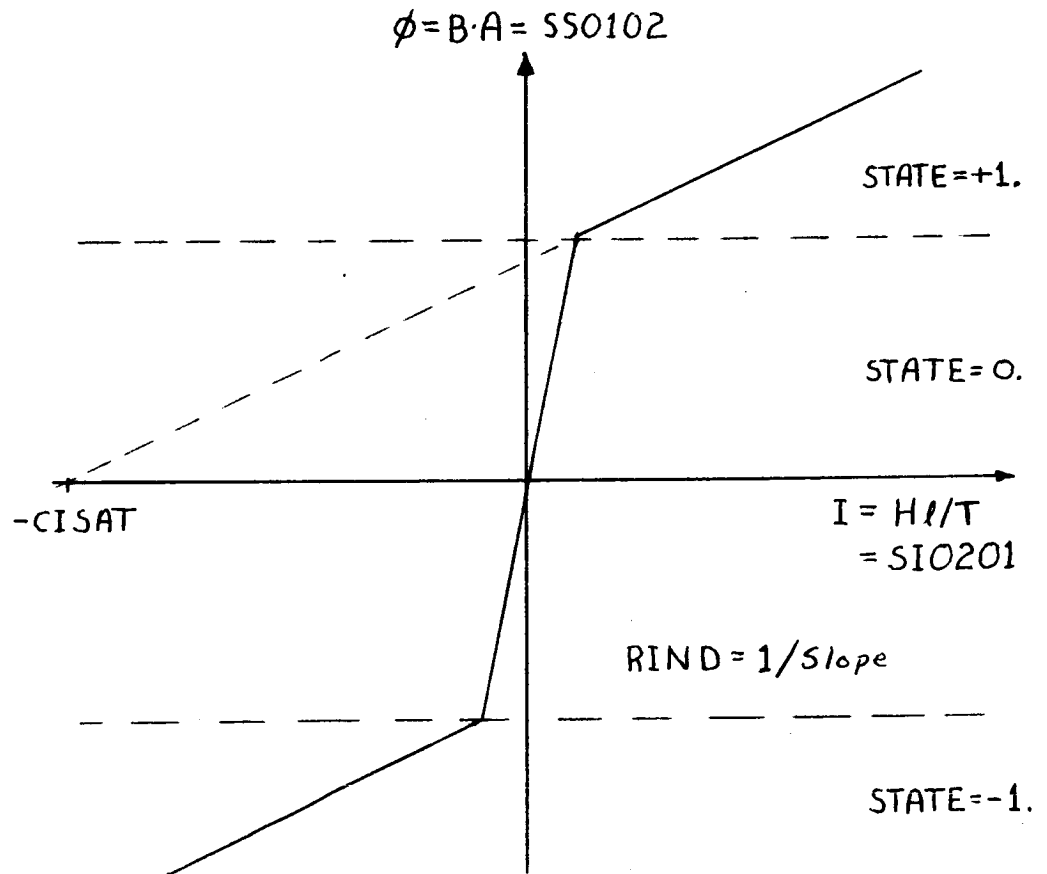
The left hand side of the first example above demonstrates the word structure of the TAG descriptors for a dependent stop function. The first character of all stop function descriptors must always be a dollar sign (\$). The second one or two characters must be some number between one and fifty which identifies the particular stop function being defined. It is this number to which the TAG control variable LLCNT is set when the value of the stop function is zero. The characters following the identification number are reserved for the name to be assigned to the stop function in the solution program. This name must follow the rules for naming Fortran variables. All dependent stop function descriptors used in the parameter description list of the TAG description deck must follow this format in order to be properly classified and dealt with by the TAG system.

The left hand side of the first example demonstrates a typical arithmetic definition of a stop function. Every stop function must have some such defining expression. This arithmetic expression may of course be of considerably greater complexity than shown here but must always depend upon either a node pair voltage or node pair voltage integral. When this expression goes to zero, the stop function sequence is triggered.

In order to keep the same stop function sequence from recurring indefinitely after it has once been triggered, a stop function should be either reset or deactivated during the stop function sequence associated with it. The sequence associated with a given stop function is identified by the fact that LLCNT equals the number of that stop function while the functions are evaluated. The stop function may be reset by adding a value to it which will drive it away from zero. Deactivation and activation of a stop function are accomplished by the subroutine TRMOD. The statement CALL TRMOD (n,0) will deactivate stop function number n. The statement CALL TRMOD(n,1) will activate stop function number n. All stop functions will normally begin in the activated state and must be deactivated if their effect is to be cancelled.

The second example, which includes both the second and third statements listed above, shows a practical useage of the dependent stop function. Two of the arguments, \$1FLXST1 and \$2FLXST2, of the subroutine CALL statement are seen to be dependent stop function descriptors. Their values are defined within the subroutine as functions of the node pair voltage integral SS0102. The values of RIND, STATE, and CISAT are changed whenever either stop function, FLXST1 or FLXST2, go to zero. These values are used in the second arithmetic statement to control the effect of voltage integral SS0102 on current source SI0201. The overall effect is to produce the piecewise linear inductance element whose terminal

properties are shown graphically below. The subroutine PLIND is listed in its entirety immediately following the illustration.



$$SIO201 = RIND * SS0102 - STATE * CISAT$$

Piecewise Linear Inductance Characteristic

SUBROUTINE PLIND(FLUX,RIND,STATE,CISAT,FLXSTI,FLXSTJ,FLXIN,DATA,
1LLCNT,LALGFT,I)

C CALL PLIND(SSXXYY,RINDI,STATEI,CISATI,\$IFLXSTI,\$JFLXSTJ,FLXINI,
C 1DATAI,LLCNT,LALGFT,I)

C SIXXXY = RINDI * (SSXXYY + FLXINI) - STATEI * CISATI

C SUBROUTINE PLIND IS A PIECEWISE LINEAR INDUCTOR CONTROL SUBROUTINE
C FOR THE TAG CIRCUIT ANALYSIS PROGRAM.

C PLIND CONTROLS THE LINEAR INDUCTOR MODEL IMPLEMENTED BY THE
C CURRENT SOURCE DESCRIPTION STATEMENT SHOWN ABOVE BY VARYING THE
C VALUES OF RINDI AND STATEI DEPENDING UPON THE FLUX LEVEL IMPRESSED
C ACROSS THE DEVICE.

C FOR FLUX LEVELS BETWEEN + AND - FLUXMX, STATE = 0. AND THE CORE
C EXHIBITS A PERMEABILITY OF UMAX YIELDING A RECIPROCAL TERMINAL
C INDUCTANCE RIND = RCPL0.

C FLUXMX CORRESPONDS TO A LEVEL OF FLUX DENSITY WITHIN THE CORE OF
C BMAX.

C FOR FLUX LEVELS ABOVE + FLUXMX, STATEI = +1. AND THE CORE EXHIBITS
C A PERMEABILITY EQUAL TO USAT = UMAX/URATIO YIELDING A RECIPROCAL
C TERMINAL INDUCTANCE RIND = RCPL1.

C FOR FLUX LEVELS BELOW - FLUXMX, STATEI = -1. AND THE CORE AGAIN
C EXHIBITS A PERMEABILITY EQUAL TO USAT AND A RECIPROCAL TERMINAL
C INDUCTANCE RIND = RCPL1.

C THE TERM - STATEI*CISAT SPECIFIES THE ZERO FLUX LEVEL MAGNITIZING
C CURRENT INTERCEPT FOR THE THREE REGIONS OF OPERATION. THIS
C INTERCEPT CURRENT EQUALS 0 IN STATE 0 SINCE THIS MODEL EXHIBITS NO
C HYSTERESIS AND -CISAT AND +CISAT IN THE +1 AND -1 STATES
C RESPECTIVELY.

C STOP FUNCTION IDENTIFICATION INTEGERS I AND J MUST BE UNIQUELY
C CHOSEN SO THAT J = I + 1 AND NO OTHER STOP FUNCTION IS IDENTIFIED
C BY EITHER OF THE SAME NUMBERS. THIS ALLOWS THE USER TO DISTINGUISH
C ALL THE VARIABLES ASSOCIATED WITH A GIVEN INDUCTOR BY APPENDING
C THE INTEGER I TO THE END OF THE NAME OF EACH ASSOCIATED VARIABLE
C AS SHOWN ABOVE. A SECOND EXAMPLE IS SHOWN BELOW OF THE CALL PLIND
C AND CURRENT SOURCE DESCRIPTION STATEMENTS AS THEY SHOULD ACUALLY
C APPEAR IN THE DEVICE DESCRIPTION PORTION OF THE TAG DESCRIPTION
C DECK.

C CALL PLIND(SS0103,RIND1,STATE1,CISAT1,\$1FLXST1,\$2FLXST2,FLXIN1,
C 1DATA1,LLCNT,LALGFT,1)

C SI0103 = RIND1*(SS0103 + FLXIN1) - STATE1*CISAT1

C ARG(1) = FLUX - TIME INTEGRAL OF VOLTAGE BETWEEN NODES XX AND
C YY IN VOLT-SECS

C ARG(2) = RIND - RECIPROCAL OF INCREMENTAL INDUCTANCE
C IN AMPS/VOLT-SEC

C ARG(3) = STATE - STATE FLAG - INDICATES PRESENT STATE OF CORE -
C -1 FOR NEG SAT - 0 FOR U=UMAX - +1 FOR POS SAT

C ARG(4) = CISAT - EXTRAPOLATED VALUE OF INDUCTOR CURRENT AT ZERO
C TERMINAL FLUX FOR STATES +1 AND -1 IN AMPS

C ARG(5) = FLXSTI - LOWER FLUX LIMIT STOP FUNCTION

C ARG(6) = FLXSTJ - UPPER FLUX LIMIT STOP FUNCTION

C ARG(7) = FLXIN - INITIAL VALUE OF TERMINAL FLUX IN VOLT-SECS

C ARG(8) = DATA - 6 MEMBER ARRAY OF CORE AND WINDING PARAMETERS

C ARG(9) = LLCNT - STOP FUNCTION FLAG - NOMINALLY EQUAL TO -1
C EQUAL TO N AT FLXSTN = 0.

C ARG(10) = LALGFT - INITIALIZING FLAG - EQUAL TO 1 ON FIRST PASS -
C EQUAL TO 2 THEREAFTER

C ARG(11) = I - LOWER LIMIT STOP FUNCTION IDENTIFYING INTEGER

C DATA(1) = PTURNS - NUMBER OF TURNS IN PRIMARY WINDING

C DATA(2) = PATHLN - MAGNETIC MEAN PATH LENGTH IN INCHES

C DATA(3) = CSAREA - MAGNETIC CROSS SECTIONAL AREA IN SQUARE INCHES

```

C DATA(4) = BMAX - MAXIMUM FLUX DENSITY IN GAUSSES
C DATA(5) = UMAX - AVERAGE MAXIMUM PERMEABILITY IN GAUSS/OERSTED
C DATA(6) = URATIO - RATIO OF PERMEABILITIES UMAX/USAT
C DIMENSION DATA(6)
C CALCULATE TOTAL TERMINAL FLUX
TFLUX = FLUX + FLXIN
IF(LALGFT-1) 100,100,110
100 CONTINUE
C CALCULATE MAXIMUM WINDING FLUX IN VOLT-SECS
C FLUXMX = 6.4516E-8 * BMAX * CSAREA * PTURNS
FLUXMX=6.4516E-8*DATA(4)*DATA(3)*DATA(1)
C CALCULATE MAXIMUM RECIPROCAL WINDING INDUCTANCE IN AMPS/VOLT-SEC
C RCPL0 = 3.1330E+7 * PATHLN / ( UMAX * PTURNS **2 * CSAREA )
RCPL0=3.1330E+7*DATA(2)/(DATA(5)*DATA(1)**2*DATA(3))
C CALCULATE SATURATED RECIPROCAL WINDING INDUCTANCE IN AMPS/VOLT-SEC
C RCPL1 = RCPL0 * URATIO
RCPL1=RCPL0*DATA(6)
C CALCULATE ABSOLUTE VALUE OF ONE STATE CURRENT INTERCEPT IN AMPS
C CISAT = RCPL0 * FLUXMX * (URATIO - 1.)
CISAT=RCPL0*FLUXMX*(DATA(6)-1.)
C CALCULATE THE ABSOLUTE VALUE OF THE BREAK POINT FLUX IN VOLT-SECS
C FLXBKP = FLUXMX
C CALCULATE THE ABSOLUTE VALUE OF THE BREAK POINT FLUX ROUND OFF
C GUARDBAND IN VOLT-SECS
DFLXBP = FLXBKP * 5.E-7
C CALCULATE ACTUAL UPPER BREAKPOINT FLUXES IN VOLT-SECS
FLUXHH = + FLXBKP + DFLXBP
FLUXHL = + FLXBKP - DFLXBP
C CALCULATE ACTUAL LOWER BREAKPOINT FLUXES IN VOLT-SECS
FLUXLH = - FLXBKP + DFLXBP
FLUXLL = - FLXBKP - DFLXBP
C DETERMINE INITIAL STATE OF CORE
IF(TFLUX-FLUXLH)10,11,11
10 STATE=-1.
GO TO 15
11 IF(TFLUX-FLUXHL)13,13,12
12 STATE=+1.
GO TO 15
13 STATE= 0.
15 CONTINUE
IF(STATE) 104,106,108
104 CONTINUE
FLUXH = FLUXLH
FLUXL = -1.E30
RIND = RCPL1
GO TO 109
106 CONTINUE
FLUXH = FLUXHH
FLUXL = FLUXLL
RIND = RCPL0
GO TO 109
108 CONTINUE
FLUXH = +1.E30
FLUXL = FLUXHL
RIND = RCPL1
GO TO 109
109 CONTINUE
C CALCULATE INITIAL VALUE OF MAGNITIZING CURRENT
FIOUT = RIND * TFLUX - STATE * CISAT

```

```

C      OUTPUT INITIAL VALUES AND CALCULATED CONSTANTS
      WRITE OUTPUT TAPE 6,1000,(DATA(I),I=1,6)
1000  FORMAT (1H1/20X,15H CORE DATA ARRAY//
120X,8HPTURNS =,E16.8,6H TURNS,14X,21HPRIMARY WINDING TURNS/
220X,8HPATHLN =,E16.8,7H INCHES,13X,21HMEAN MAG. PATH LENGTH/
320X,8HCSAREA =,E16.8,10H INCHES**2,10X,20HCROSS SECTIONAL AREA/
420X,8HBMAX =,E16.8,8H GAUSSES,12X,20HMAXIMUM FLUX DENSITY/
520X,8HUMAX =,E16.8,16H GAUSSES/OERSTED,4X,20HMAXIMUM PERMEABILIT
6Y/20X,8HURATIO =,E16.8,8H (RATIO),12X,21HRATIO OF UMAX TO USAT)
WRITE OUTPUT TAPE 6,1001,FLUXMX,CISAT,RCPLO,RCPL1
1001  FORMAT (1H0/ 20X,31H CALCULATED INDUCTANCE CONSTANTS//
120X,8HFLUXMX =,E16.8,10H VOLT-SECS,10X,53HSATURATION FLUX LEVEL -
2ALSO EQUAL TO BREAKPOINT FLUX/
320X,8HCISAT =,E16.8,5H AMPS,15X,54HMAGNITUDE OF FLUX AXIS INTERCE
4PT CURRENT IN SATURATION/
520X,8HRCPLO =,E16.8,14H AMPS/VOLT-SEC,6X,39HRECIPROCAL INDUCTANCE
6 FOR HIGH U REGION/
720X,8HRCPL1 =,E16.8,14H AMPS/VOLT-SEC,6X,42HRECIPROCAL INDUCTANCE
8 FOR SATURATED REGION)
WRITE OUTPUT TAPE 6,1002,TFLUX,FIOUT,STATE
1002  FORMAT(1H0,/20X,27H INITIAL STATE OF INDUCTANCE//
120X,8HTFLUX =,E16.8,10H VOLT-SECS,10X,30HINITIAL VALUE OF TERMINA
2L FLUX/
320X,8HIMAG =,E16.8,5H AMPS,15X,36HINITIAL VALUE OF MAGNITIZING C
4URRENT/
520X,8HSTATE =,F5.1,31X,21HINITIAL STATE OF CORE/ )
      GO TO 120
110 IF(LLCNT - (I + 1)) 111,111,120
111 IF(LLCNT - I) 120,130,140
120  CONTINUE
C      CALCULATE VALUES OF FLUX LIMIT TRIGGER FUNCTIONS
      FLXSTI = + TFLUX - FLUXL
      FLXSTJ = - TFLUX + FLUXH
      RETURN
130  IF(STATE) 150,160,170
150  GO TO 220
160  CONTINUE
      FLUXH = FLUXLH
      FLUXL = -1.E30
      STATE = -1.
      RIND = RCPL1
      GO TO 120
170  CONTINUE
      FLUXH = FLUXHH
      FLUXL = FLUXLL
      STATE = 0.
      RIND = RCPL0
      GO TO 120
140  IF(STATE) 170,190,200
190  CONTINUE
      FLUXL = FLUXHL
      FLUXH = +1.E30
      STATE = +1.
      RIND = RCPL1
      GO TO 120
200  GO TO 230
220  WRITE OUTPUT TAPE 6,1010
1010  FORMAT (1H0,33H LOWER TRIGGER FIRED IN REGION -1.)
      GO TO 120

```

230 WRITE OUTPUT TAPE 6,1020
1020 FORMAT (1H0,23HUPPER TRIGGER FIRED IN REGION +1.)
GO TO 120
END

D. Requirement and Usage of Fortran Specification Statements

1. General Considerations

The TAG description deck may require the use of Fortran II statements in both the parameter description list and the output and control sequence (to be described in the next section). In order to use the full capabilities of this language it is necessary to understand and use the so-called Fortran specification statements. For a complete list and detailed explanation of these statements, the previously mentioned Fortran manuals are suggested as adequate reference material. The names of these statements are DIMENSION, COMMON, EQUIVALENCE, and FREQUENCY. The DIMENSION statement is the only one that will be described here since it is the most frequently useful of the four and is always required in the TAG description when plotted outputs are called for.

The specification statements are provided by Fortran for the general purpose of increasing the efficiency of a given program by reducing the amount of memory required, or by reducing the number of basic machine operations required, or by streamlining the program flow. Fortran arithmetic and subroutine statements dictating the usage of specification statements will occur in both the parameter description list and output and control sequence list. Regardless of their origin, all Fortran specification statements which appear in the TAG description deck should follow immediately after the parameter list.

Even though the TAG description deck borrows a part of its word and statement structure from Fortran II, it should not be considered a Fortran program itself. For this reason the sequence rules stated for the TAG description deck always take priority over those stated for Fortran programs.

2. The DIMENSION Statement

Example: DIMENSION BFI (100), DATAT(17)

In preparing Fortran statements it is often desirable to apply a single name to a whole list of variables or parameters. A set of variables or parameters referred to by a single name is obviously multiple-valued, and storing it in the computer memory requires as many word locations as there are members of the list. The DIMENSION statement is required by Fortran to list the number of memory locations to be allocated to each multi-valued variable or parameter. This number is listed as an integer in parentheses following the variable name and must be equal to or larger than the number of values referred to by that name. Any variable name not appearing in a DIMENSION statement will automatically be assigned a single memory location in which to store its value.

The variable names which appear in DIMENSION statements are separated from one another by commas as shown in the above example. The spacing of variables on the card is at the complete discretion of the user since blank

spaces are ignored by Fortran. Entries may continue from card to card following the rules for Fortran statement continuation. Any number of DIMENSION statements may be used to cover all multi-valued variables appearing in the TAG description deck.

The example shows a typical two entry DIMENSION statement as it might appear in TAG. The first entry, BF1, is of the type that is always required when a plotted output is desired. The TAG automatic plot routine requires a set of 100 memory locations to be named and made available to it for each two variable output curve specified. If more than one output plot is desired, additional memory buffers must be dimensioned each for 100 locations. These could be added to the DIMENSION statement as BF2(100), BF3(100), etc.

The second entry in the example, DATAT(17), might represent a list of 17 parameters required in the calculations performed within a device model subroutine. Such a subroutine would be named in a CALL statement appearing in the parameter description list, and the name DATAT would appear as one of its arguments.

E. The TAG Output and Control Sequence and Description Deck
End Statement

1. General Characteristics of the Output and Control
Sequence

a. Content of the Output Sequence

The listing of the output sequence follows immediately after the Fortran specification statements and controls the output of all circuit variables whose values are required by the user to display the results of the simulation process. Output variable specification and labeling, print-out format specification, and automatic curve plotting are called out by the user within this sequence. In addition to pure output specification, any desired Fortran coding can be added to this part of the TAG description deck for the purpose of computing secondary results, summaries of results, special control functions or any other desired expressions. In section D, only the print-out and plot-out specification statements will be described in detail since their use is almost universally required.

b. Entering the Output Sequence

The first statement of the output sequence must contain a statement number in columns one through five of the card on which it appears. This number is used in the TAG solution program as the link between the internally computed circuit equations and the users output sequence. As such, its existence

is absolutely essential; furthermore, it must be the first statement number to appear in the TAG description deck. Numbers larger than 5999 should not be used as statement numbers anywhere in the output sequence because they may already be assigned to the solution program by the TAG processor.

For pure D.C. steady state problems, the output sequence is executed only after a solution is achieved which meets the criteria specified within TAG or by the user for node current equilibrium. For all transient problems the output sequence is executed periodically at a rate specified by the value of the TAG control variable FØUT. FØUT is the period in simulated time between succeeding executions of the output cycle. The number assigned to FØUT by the user specifies the time resolution of the transient simulation results.

In addition to this purely periodic coupling between the output sequence and the internal computational sequence, the user can force the program to execute the output sequence at either a specific value of a circuit dependent variable or a specific point in simulated time. The circuit variable controlled link occurs at the exact point at which a dependent variable stop function is satisfied. With the circuit in the exact state specified by the stop function, the program executes a single pass through the output sequence. Stop functions are specified within the dependent parameter list.

The time controlled link is achieved by the presences of UTF functions in the parameter description list. When simulated time becomes equal to the argument of a UTF function, the program directs the output sequence to be executed twice at the exact time specified. The first pass occurs with the value of UTF equal to one and the second occurs with the value of UTF equal to zero.

c. Ending the Output Sequence

The only link between the user and the solution of the simulation equations is through the output sequence. It is therefore within this sequence of operations that the user must exercise control over the overall simulation process. This means that, by the end of the output sequence, the decision must be made whether to continue the present simulation run, change values and start a new simulation run, halt the simulation all together, or take some other desired action. The final statement of the output sequence must direct the program to the next sequence of operations to be performed.

Decision making and program control are accomplished using the Fortran control statements described in the referenced Fortran manuals. Section 5.0 of this report will describe in detail the flow of a typical solution program. This information is necessary for a complete understanding of solution process control. However, a few examples of typical output

sequence endings will be provided here for the beginner to follow.

1) Single Pass D.C. Steady State Solution

Example: ~~GOTO~~ 6000

The last statement of the output sequence would be as shown in the example above. After all the steps of the output sequence were performed, the program would switch to the internal TAG statement numbered 6000 and proceed. In all TAG solution programs the sequence of events started by statement 6000 has the effect of resetting the solution process, reading in a new set of parameter data, and starting a new solution run. If there is no new parameter data to be read, the program is automatically terminated.

2) D.C. Steady State Transfer Functions for a Stepped Input

Example: $SV0001 = SV0001 + DELTAV$
IF(SV0001-V01MAX) 62000,6000,6000

This setup has the effect of producing a D.C. steady state solution for every value of $SV0001 + SV0001(\text{initial}) + N * DELTAV$ (for $N=0,1,2 \dots$) up through $SV0001 + V01MAX$. The values of $SV001(\text{initial})$, $DELTAV$, and $V01MAX$ are supplied by

the user in the data list. The IF statement controls this process by routing the program to statement number 6200 as long as the value of SV0001 is less than or equal to V01MAX. At statement number 6200 the solution process is begun again using the new value of SV0001 and the previous solution for its set of initial values. When SV0001 becomes greater than V01MAX, the process is terminated by routing the program to statement number 6000 which has the same effect as explained in the previous example.

3) Transient Simulation Without Plotted Outputs

Example: IF(FT-DØNE) 6200,6000,6000

For transient simulation the output sequence is automatically executed each time simulated time, FT, reaches an integral value of the time resolution control variable FØUT. The effect of the above output sequence terminating statement is to continue the simulation run as long as FT is less than the value of the user supplied termination variable, DØNE, by directing the program to statement number 6200. When FT becomes equal to or larger than DØNE, the simulation run is terminated by directing the program to statement number 6000. By this simple method termination may be controlled to no finer resolution than FØUT since the test for termination is made within the output sequence. This can of course be remedied, if required, by using

the statement UTF (DØNE) somewhere in the parameter description list.

2. Automatic Print-Out Specification

```
Example:  TIME   = FT           D
          VR1    = SV0304       S
          IR1    = SV0304*SG0303 S
          VCBQ2  = SV0204       S
```

TAG provides a simplified format for specifying, naming, and formatting output variables to be printed. A simple Fortran equality statement with a print control character D, S, or E in column 73 of the statement card is all that is required. TAG takes these automatic print-out statements and produces the proper Fortran WRITE ØUTPUT TAPE and FØRMAT statements for the solution program.

The output will appear as a single column of output variable names equated to their numerical values. The only control that can be exercised over this format is to vary the vertical spacing between adjacent outputs by specifying either E, D, or S as the print control character. If the automatic print-out format supplied by TAG is not desired, the user can place his own Fortran output statements in the output sequence list. Any arithmetic statement appearing in the output sequence which contains no print control character in column 73 will be treated in the solution program as a normal Fortran statement and will have no WRITE ØUTPUT TAPE or FØRMAT statement generated for it.

The name that will identify a particular quantity in the printed output list is supplied by the user as the variable appearing to the left of the equal sign in an automatic print-out statement. Such names should be chosen to convey to the user the nature of the quantity they identify. They may be identical to variables appearing in the TAG equivalent circuit or arbitrarily chosen by the user. In any event, the names chosen must follow the rules for naming Fortran floating point variables and must not, therefore, begin with numbers or the letters I, J, K, L, M, or N. All output values are expressed in floating point decimal notation to eight significant figures.

The right hand side of the equal sign of an automatic printout statement is reserved for specifying the quantity which the name on the left is to identify. This quantity must be expressed in the same language that was developed to describe the TAG equivalent circuit to the equation generator. Since the TAG equations are written in terms of node pair voltages, such voltages may be specified directly using the same type of description and polarity rule as was used to form the connection list. This is shown in the first and third examples above. Quantities not calculated directly by the simulation equations or parameter description statements may be calculated either separately in the output sequence or as part of the automatic printout statement. This is shown in the second example above. The quantities normally available for direct printout are node pair voltages, node pair voltage integrals,

simulated time, and all element parameter values.

The print control character must be placed in column 73 of any arithmetic statement whose value is to be printed out. In addition to flagging a particular statement as an automatic printout statement, the print control character serves to control the vertical spacing of the output listing. S stands for single space and then print. D stands for double space and then print. E stands for eject (reset to the top of the next page) and then print.

It should be noted that, even though IBM cards have eighty columns, standard Fortran coding sheets only show 72. To overcome this deficiency a special note should be made to the key punch operator which clearly labels the right hand margin of the coding sheet as column 73. The print control characters are then placed in this margin area in line with the statements to which they belong. This will be illustrated in the examples at the end of this section. If a multi-card arithmetic statement (which must follow the rules for Fortran continuation) is required as an automatic print-out statement, a print control character should appear in column 73 of each card.

The example shown above will produce the following printed output. This list will continue for all values of $TIME = N * F\emptyset UT$ ($F\emptyset UT = .5$ and $N = 0, 1, 2, \dots$) up to termination time, $D\emptyset NE$ (see section E, paragraph 1-c).

TIME = 0.00000000E-00
VR1 = 0.
IR1 = 0.
VCBQ2 = 0.

TIME = 0.50000000E 00
VR1 = 0.
IR1 = 0.
VCBQ2 = 0.

TIME = 0.10000000E 01
VR1 = 0.
IR1 = 0.
VCBQ2 = 0.

3. Automatic Plot-Out Specification

The TAG system offers automatic two variable curve plotting as one of its convenience features. This is accomplished by a special set of subroutines which have been made a part of the system and are accessible to the user through a single relatively simple CALL statement. The statement CALL SCØPE (--), supplied with a suitable set of arguments, is all that the user need supply to the output sequence for each plot desired. The program will automatically collect and store the desired data points, scale and label the axes, properly format this information, and output it onto a special magnetic plot tape for off-line processing on a Stromberg-Carlson 4020 plotter.

The plot subroutine CALL statement is set up like any Fortran CALL statement and requires 7 arguments supplied to it by the user. The definition and order of the arguments are shown below along with a sample CALL SCØPE (--) statement.

```
CALL SCØPE (X, Y, BFN, M, K, IHYname, JHXname)
```

- Where
- X is the name of the independent variable
 - Y is the name of the dependent variable
 - BFN is the name of a group of memory locations dimensioned by the user and available to the scope routine. BFN should be dimensioned at least 100.
 - M is the number of memory locations allocated for BFN (normally 100)
 - K is a control variable.
K must be equal to 1 for collecting data points and equal to 2 on the final pass through the plot routine at which time the axes are scaled and labeled.
 - IHYname is a Hollorith specification for the dependent axis label.

In this specification I is the number of characters which appear in the Yname label (including all blanks and punctuation except for the final comma). H is the character H which denotes a Hollerith specification.

- Xname is the group of characters which make up the independent axis label.

Example: CALL SCØPE (FT,SC0103,BF1,100,K,9HVBE VØLTS,
9HTIME SECS)

The particular automatic plot statement shown in the above example specifies a plot of the node pair voltage, SV0103, as a function of simulated time, FT. Both the time and voltage axes of the final plot will appear properly scaled and will be labeled "TIME SECS" and "VBE VOLTS" respectively. Unfortunately this single CALL statement will not by itself produce the plot since it does not provide the required control over the variable K. K must be controlled by the user to equal 1 during normal data gathering passes and 2 during the plot finalization pass. Two methods by which the user may achieve this control are illustrated below.

a. Repeated CALL statement

```
CALL SCØPE (X,Y,BF1,100,1,LH1,LH1)
CALL SCØPE (W,Z,BF2,100,1,LH1,LH1)

IF(FT-DØNE) 6200,60,60

60 CALL SCØPE(X,Y,BF1,100,2,1HY,1HX)
   CALL SCØPE(W,Z,BF2,100,2,1HZ,1HW)

GØ TØ 6000
```

In the repeated CALL statement method, illustrated above, the output sequence is routed through only the first set of CALL statements (in which K equals one) as long as FT is less than DØNE. When FT

becomes greater than or equal to DØNE, the second set of CALL statements (in which K equals two) is executed just prior to process termination. As many plot statements as desired may be accommodated by this method. The axes names supplied in the first set of CALL statements are not used and may therefore differ from those appearing in the second set of CALL statements.

b. Explicit control of K

```
      K = 1
      IF(FT-DØNE) 61,60,60
60    K = 2
61    CALL SCØPE (X,Y,BF1,100,K,1HY,1HX)
      CALL SCØPE (W,Z,BF2,100,K,1HZ,1HW)

      IF(K-2) 6200,6000,6000
```

In this method the CALL statements are coded only once. K appears explicitly as the control variable in each. The first IF statement controls the value of K used by the CALL statements. For FT less than DØNE statement 60 is skipped over and the CALL statements are executed with K equal to 1. The second IF statement continues the simulation process by routing the program to TAG statement number 6200. When FT becomes equal to or greater than DØNE, K is set to two by statement number 60 before the CALL

statements are entered. The plots are then finalized with K equal to two. The simulation process is then terminated by the second IF statement which routes the program to TAG statement number 6000. When many plots are desired, this second control method has an obvious advantage in coding efficiency.

4. The TAG Description Deck END Statement.

Example: END

The final statement in the TAG description deck must be a Fortran END statement. This notifies the TAG solution program generator that the last card in the TAG description deck has been reached. Such an END card is always necessary.

5. Example Output Sequence and END

The following example output sequence coding sheet refers to the example circuit at the end of section 3-C. The desired output variables are input voltage, input current, transformer primary voltage, current in inductor L_p , transformer secondary voltage, voltage across load resistor R_L , and current through R_L . It is desired that TAG produce a time history of the response of these variables from FT, the time variable, equal to zero to FT equal DONE, a variable whose value is entered in the data list at execution time. It is further desired that each block of output information be identified by the time at which it occurs, and be separated from other blocks of data by a double space. Both a printed output listing and a set of plotted output curves are required.



FORTRAN CODING FORM

Program <u>Test Circuit #1 - Output Sequence</u>		Punching Instructions										Page <u>2</u> of <u>2</u>	
Programmer <u>J.O.P.</u>		Date										Card Form #	Identification 73 ————— 80

C FOR COMMENT

[illegible]

F. Set-Up of the TAG Data Deck

1. General Characteristics of the TAG Data Deck

The TAG data deck is distinguished functionally from the TAG description deck by the fact that it supplies circuit parameter and simulation control information to the solution program rather than to the TAG program generator. All significant, non-zero, variable parameters and control constants named in the description deck must be supplied with numerical values which will allow the solution program to perform the desired circuit simulation.

TAG provides, as part of the solution program, a special input subroutine which allows the user to list all numerical data in a simple but flexible format. Each number appearing in the data list is identified by the name of the variable to which it belongs. During the generation of the solution program, the input subroutine is supplied a list of all the variable names in the TAG description deck specified by the user as well as several internal TAG control variables. When the solution program is executed, this subroutine matches the names in its variable list to the names in the data deck and assigns the indicated numerical values to the proper program variables. Therefore, the names in the data list must correspond exactly to the names which appear in the TAG description deck. This applies particularly to the order of the node numbers in a branch element descriptor.

In addition, the input routine provides for both commenting and limited carriage control of the data list which is reproduced during program execution as a check for the user. Any data card in which a C appears in column 73 is treated by the input routine as a comment card and is therefore not processed as data. This allows the user to label each data set and execution run as desired to avoid confusion. An E appearing in column 74 will cause the input routine to instruct the printer to eject a new page before continuing the data list printout.

2. Data List Format

Example:

```

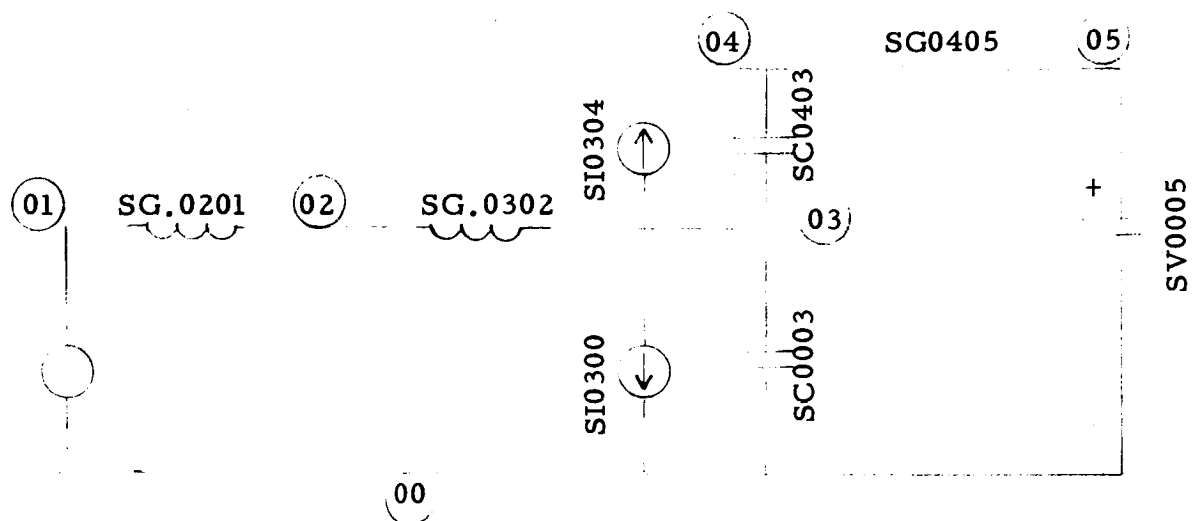
TRANSISTOR INVERTER TEST CIRCUIT                                CE
SV0001 = 10., SV0005 = 20.,
SC0201 = 0.1E-9,
SG0201 = 0.1E-3, SG0405 = 1.0E-3,
SG0302 = 100.E-3,
DATAT = .075E-12, .149E-12, 0.0, 0.0, .991, .500
        2.0E-9, 2.0E-7, .75, .75, .50, .333,
        95.E-12, 33.E-12 0.0, 0.0, .026,
DONE = 1.0E-6, FOUT = 0.01E-6,
SPEED-UP CAPACITOR 0. VOLT INITIAL CONDITION                    C
SV0201 = 0.0, SV0403 = -20., SV0003 = 0.0, *
SPEED-UP CAPACITOR 5.VOLT INITIAL CONDITION                     C
SV0201 = -5., SV0403 = -25., SV0003 = 0.0,*

```

The example data list shown above might be considered typical for the simple transistor inverter circuit whose TAG equivalent circuit is shown below. Both single and multiple valued data statements

are shown. The final two statements specify the capacitor initial conditions for two separate simulations. Values for the two transistor current sources and shunt capacitances are calculated by the users transistor subroutine from parameter values given in the list DATAT. The DATAT list given here is an approximate set of parameter values for the 2N910 transistor modeled by the TRAN3 subroutine listed in section III.C.5.b.

Example:



TAG Equivalent Circuit of Example Transistor Inverter

The data list is formed as a series of data statements. Each statement equates a particular variable name to the one or more values which belong to that name. While many of these statements look like simple FORTRAN arithmetic statements, they should never be confused with actual FORTRAN coding.

Each data list statement contains a variable name, an equal sign, and a list of one or more numerical values. The statements are punched anywhere in columns one through seventy-two of an IBM card. As many statements may be placed on a single card as desired. Blank spaces may be left anywhere within these statements to augment clarity and readability. The statements are separated only by a comma and the last statement of a given list is followed by an asterisk. This makes the data list format identical in punctuation to that of the connection list. The statements follow from card to card and may use as many cards as are required to specify the values of all variables appearing in the solution program.

The values for a multiple valued variable follow the equal sign as a list of numbers separated by commas. Such a list may follow from card to card without change of format for as many cards as required. FORTRAN continuation notation is not required and must not be used here. The end of the list is detected by either the next variable name or the finalizing asterisk.

Unlike the connection list, the last data statement may be followed by a comma before the finalizing asterisk. This allows for standardization of all data statements for ease in data list modification. The asterisk may then appear by itself on the final card.

3. Multiple Data Lists

As many different sets of data may be supplied to the solution program for a single computer run as desired. This is accomplished by stacking the data lists one behind the other separated only by the finalizing asterisk of each. If the final statement of the output and control sequence routes the program to TAG stage number 6000 after each simulation run, the data lists are processed one at a time in their order of occurrence in the data deck. After the last list is processed, the program is automatically terminated.

If the data lists for a given computer run differ from one to the next by only a few values, a complete listing of data values need be given only once in the very first list. Each subsequent list is completely specified by the data statements whose magnitudes actually change. The program will always remember the previous data list values and substitute the new ones only where required.

In the example data deck of the previous section two data lists are shown. The first list includes all the data statements down to the first asterisk. This list completely specifies all values required

for a successful simulation run. The second data list, which ends at the second asterisk, implicitly includes all the entries in the first data list except for the single explicit initial condition voltage statement shown. The solution program is first executed using the parameter values specified in the first list. It is then run again with only the parameters of the second list changed in value.

4. TAG Internal Control Parameters

TAG has built into it a set of internal variables and control constants which are assigned nominal values by the solution program. A subset of these are available in the input list and may be controlled by the user if the normally supplied nominal values are not suitable for a given simulation. The two distinct groups into which this set of controllable constants fall are transient simulation control variables and non-linear DC steady state control variables. Simulations containing a mixed set of algebraic and differential equations will involve both sets of controls.

a. Transient Simulation Control Variables

FT-Simulated Time.

The name used by TAG for the independent variable, time, is FT for all transient runs. This is available in the input list for the purpose of allowing simulated time to be started at any initial value desired. If such a value is not supplied to the data list, the execution program will always start at initial time $FT = 0$.

FØUT - Time increment between output sequence executions.

The variable FØUT is used by the TAG simulation program to effect periodic execution of the users output and control sequence. Whenever FT equals $FT(\text{initial}) + n * FØUT (n=1,2,---)$, the integration process is stopped and the output and control sequence is executed. Such a stop is called a print stop since its usual purpose is to output information to the user. Reentry from a print stop into the internal integration sequence does not accommodate discontinuities in the simulation equations. Therefore, a print stop may not be used to change the values of any parameters which effect the simulation equations. If FØUT is not supplied a value by the user, it will be set to zero. A zero value of FØUT will cause the simulation program to execute the output sequence at UTF stops and dependent variable stops only. If there are no UTF stops or if all UTF stops have been executed the output sequence will be executed every $10 * FSTEP$. The user should normally provide a value of FØUT to the data list.

DØNE - Simulation termination control.

The termination control variable is supplied to the program by the user and is included here only to complete the list of normal transient simulation control variables. If the user controls termination in the normal manner by including a statement IF (FT - DØNE) 6200,

6000, 6000 at the end of the output and control sequence, the variable DØNE must be assigned a value of time and included in the DATA list. If such a value is not provided by the user, the program will assume DØNE equal to zero. The user may, of course, use any variable name desired other than DØNE for termination control as long as it is unique within the program and does not start with letters I, J, K, L, M, or N.

FSTEP - Initial integration time step.

FSTEP is the initial value of time step used by the transient simulation equation integrator. In the Adams-Moulton variable step size integration mode the integration step size is controlled internally as a function of the allowed truncation error. However, the process is started in the Runge-Kutta mode using fixed-step size equal to FSTEP. After a sufficient number of points have been calculated the process switches to the Adams-Moulton mode for increased computational efficiency. The step size will then be halved or doubled depending upon satisfaction of certain local error control criteria. In the Runge-Kutta integration mode integration remains in the Runge-Kutta starting mode for all time and the step size is maintained at the value of FSTEP. The simulation program sets FSTEP to 1.E-11 unless a different value is entered into the data list by the user. FSTEP should be smaller than the smallest local circuit time constant. If UTF functions are used, FSTEP should be larger than

FT *1.E-8 for any value of FT at which such a stop occurs.

LTYPE - Integration mode selector.

LTYPE is an integer variable and its value is therefore listed without a decimal point. LTYPE is normally set to 4 by the simulation program. For LTYPE equal to 4 the integrator is set to the Adams-Moulton variable step size mode. The user may set LTYPE equal to 2 which will force the integrator to remain in the fixed step size Runge-Kutta mode.

FEPSL2 - Relative lower bound on truncation error.

In the Adams-Moulton variable step size mode, when the measure of the local truncation error drops below FEPSL2 times the value of the dependent variable for any equation being integrated, the integration step size is automatically doubled. FEPSL2 is nominally set by the execution program to equal 5.E-6.

FEPSL3 - Relative upper bound on truncation error.

In the Adams-Moulton variable step size mode, when the measure of the local truncation error rises above FEPSL3 times the value of the dependent variable for any equation being integrated, the integration step size is automatically halved. FEPSL3 is nominally set by the execution program to equal 5E-4.

FEPSL4 - Smallest allowed integration step size.

In the Adams-Moulton variable step size mode FEPSL4 is the smallest value of integration step size allowed. FEPSL4 is nominally set by the execution program to equal 1.E-16. FEPSL4 should always be set at least an order of magnitude below the smallest expected local circuit time constant.

b. Non-Linear DC Steady State Control Variables

FEPSL - Maximum relative dependent variable step size at a solution.

The Newton-Raphson process is used to solve the non-linear algebraic simulation equations which arise in the characterization of circuits and segments of circuits which do not include either capacitive or inductive elements. The Newton-Raphson process attempts to move the state of the circuit from an initial state which is not at equilibrium (as defined by Kerchoff's current and voltage laws) to the equilibrium state corresponding to the specified energy source distribution. This may be viewed as choosing a tree branch voltage vector $V]$, which by definition satisfies Kerchoff's voltage law, in such a way as to minimize a residual function $F(V])$ which is a measure of the sum of the currents into each circuit mode. As the residual function $F(V])$ is forced toward zero, the sum of the currents into each node will also be forced toward zero, thus achieving increasingly better satisfaction of Kerchoff's current law. The algorithm which is used to accomplish this is as follows:

$$V_{n+1}] = V_n] - \frac{F(V_n])}{\frac{d F(V_n])}{d V_n]}}$$

- where:
- V_{n+1} is the tree voltage vector generated at the nth iteration step.
 - $F(V_n])$ is the residual function evaluated using tree voltage vectors V_n which was generated at the (n-1)th step.
 - $F(V_n])$ is the matrix of derivations of the residual functions with respect to the tree voltage vector $V]$ evaluated at $V] = V_n]$.
 - $V_o]$ is the tree voltage vector which defines the initial (non-equilibrium) state of the network.

TAG requires that the largest member of $V_{n+1}] - V_n]$ be less than $FEPSL * V_n]$ before the process may be said to have converged to a correct solution. The simulation program sets FEPSL to a nominal value of 5.E-6.

FEPSL1 - Maximum relative residual at a solution.

In addition to the step size criteria explained above, TAG requires that the largest value of the residual vector $F(V_n])$ be less than $FEPSL1 * V_{n+1}]$ before the process may be said to have converged to a correct solution. The simulation program sets FEPSL1 to a nominal value of 5.E-6.

FEPSL1 - Maximum relative residual at a solution.

In addition to the step size criteria explained above, TAG requires that the largest value of the residual vector $F(V_n)$ be less than $FEPSL1 * V_{n+1}$ before the process may be said to have converged to a correct solution. The simulation program sets FEPSL1 to a nominal value of 5.E-6.

LMAX - Maximum allowed number of steps to achieve a solution.

The maximum number of Newton-Raphson steps allowed to achieve solution convergence is set by the value of LMAX. This control is required, because in many situations in the present formulation of TAG, solution convergence will never be achieved unless the initial circuit state is sufficiently close to the final solution state. LMAX will limit the number of iterations so that unnecessary waste of computer time will be avoided. The simulation program sets LMAX equal to 50. This is an integer constant.

LDBG01 - Diagnostic print control variable.

When convergence of the Newton-Raphson process is not achieved in LMAX number of step, it is often helpful to print out the results of each step. The integer constant LDBG01 is provided to allow such a diagnostic print out. When the internal step counter passes the value LDBG01, the number of each equation, the value, F,

of each residual, and the value, X, of each tree branch voltage is printed out. The equations are in the same order as listed in the Fortran solution program. The solution program sets LDBG01 to 51 which is one greater than LMAX and insures that, for LMAX equal to 50 no diagnostic print out will occur.

4.0 TAG SYSTEM DECK SET-UP

A. General Comments

The IBM card deck that is actually submitted to the computing system for execution is herein called the TAG system deck. The system deck may be submitted in any one of several forms depending on the requirements of the user. Most of these forms are presented in part B of this section.

The TAG system deck is composed of several smaller subsystem decks stacked together to form one large deck. Some of these smaller decks, like the TAG description and data decks, are unique to a given circuit and must be supplied by the user. Some of the other small decks supply the machine coding for the general solution and control subroutines required during execution of the TAG generated simulation programs and are therefore a permanent part of the TAG execution system. A third type of subsystem deck is the component modeling subroutine which may start out as user supplied coding for a particular circuit simulation but may be incorporated into the permanent execution deck if applicable to many circuits.

Each of these subsystem decks will be written in one of the five languages provided for in the TAG - FORTRAN II system. These five languages are:

- TAG - Used in the TAG description deck only
- FORTTRAN - Used in component modeling or control subroutines
- FAP - May be used like FORTRAN but not recommended for general use by engineers.

- BINARY** - Used for all permanent parts of the system execution deck. These binary decks are produced by the system compilation of all FORTRAN and FAP source decks.
- DATA** - The language of the data deck is explained in section 3.F and is unique to the TAG system input subroutine.

Regardless of the form of the system deck or the purpose of a computer run, certain basic system rules must be followed to attain proper results.

1. Regardless of the functions of a particular subsystem deck it must be ordered in the TAG system deck according to its language content. All such decks will appear following the system control cards. The language order is as follows: TAG, FORTRAN, FAP, BINARY, DATA.
2. The first IBM card in any deck must be a brown \$JOB card. Its format and contents are illustrated below.

```
      A , B      , C      , D , E ,F, G
$JOB  CIB,5502000,40007-0,78877,AFC1,5,10000
```

- A - Programmer initials
- B - Program identification number
- C - Work order number
- D - JPL employee number
- E - Run specification information
 - A - Bld 125
 - F - Fortran II Monitor
 - C - Code check
 - 1 - Single copy listing

F - Estimate of run time in minutes

G - Estimate of lines of printing in output listing

The rest of the card may be filled in by the user with any identifying symbol or name.

3. The last card in any deck must be a Green EOF or End of File card. The EOF card has 7 and 8 punch in column 1. EOF cards may be obtained from the computing system office.
4. Each time a deck is submitted to the computing system for a computer run, a job request form must be filled out and submitted with it. An example form filled out for a typical TAG combination compilation and simulation run is shown below. These forms are supplied by the computing system office.

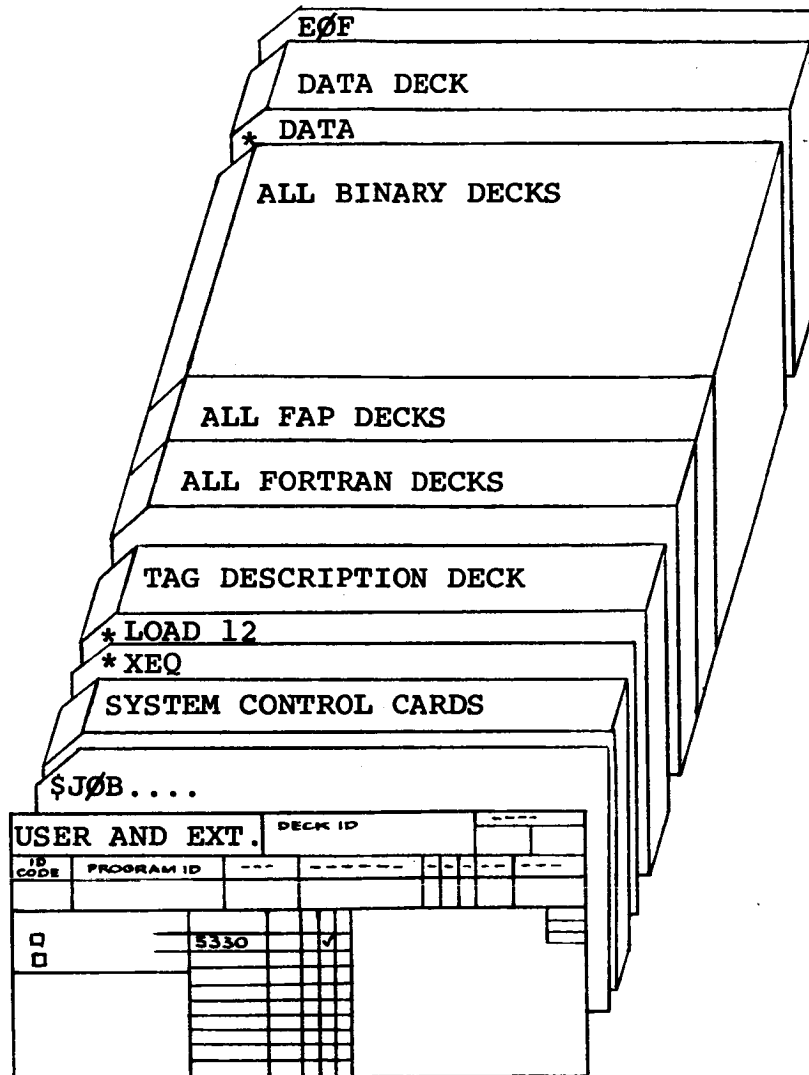
USER AND EXT				DECK ID				OUTPUT DISTRIBUTION			
								BLDG.		BOX	
<i>CIB 5332</i>				<i>5502</i>				<i>125</i>		<i>63</i>	
ID CODE	PROGRAM ID	WORK ORDER NO.	EMPLOYEE NO.	AREA	SYS	STA-TUS	FORM	TIME EST	LINE COUNT		
<i>CIB</i>	<i>55020000</i>	<i>40007-0</i>	<i>78877</i>	<i>A</i>	<i>F</i>	<i>C</i>	<i>1</i>	<i>5</i>	<i>10000</i>		
PLOT FRAMES <input type="checkbox"/> SAVE <input type="checkbox"/> TAPE <input type="checkbox"/> ROTATE <input type="checkbox"/> TUBE		F80 <i>5</i> 35MM	TAPE NO. <i>5330</i>	ASSIGNED TO <i>WJT</i>	DENSITY 200 556 800 <input checked="" type="checkbox"/>		COMMENTS <i>NØ RING</i> <i>PLØT 5 FRAMES ØN SC4020</i>				R S E T A A U U N U S S
OPERATIONS USE ONLY											
TIME OUT											
TIME IN											
LOAD TERMINAL			MOVE TO								
LOAD SEQ			PLOT TAPE NO								

SCF DCOS REQUEST

JPL 1102 FEB 66

5. The card immediately preceding the TAG description deck must be a * LOAD 12 card. This causes the TAG preprocessor to be read into the computer from tape and surrenders control of the machine to it. In no instance should a card appear between the * LOAD 12 card and the first card of the TAG Description Deck connection list.
6. The card immediately preceding the data deck must be a * DATA card.
7. For a simulation run to be made either as part of a TAG compilation run or as a subsequent normal execution run using the TAG generated simulation program in either its FORTRAN or Binary form, a * XEQ card must be included in the deck between the system control cards and the start of the actual program cards.
8. Between the \$JOB card and the first Fortran monitor control card (such as * XEQ or * LOAD 12) the system control cards are placed. These cards generally setup and assign tape units for input-output requirements. Scratch tapes need only an ASSIGN card. Actual input or output tapes require both a SETUP and an ASSIGN card for each. The ATTEND 0,77777 card provides for a system core dump in the event that either the output line count or run time estimates are exceeded. Additional information about the system control cards can be acquired at the computing system office.

1. General One Pass Preprocessor and Simulation Run



Pictorial representation of the most general form of the TAG system deck.

The most general form of the TAG system deck occurs in the one pass combination preprocessor-simulation run. In this mode of operation a full set of subsystem decks and control cards are required. Such a deck is shown pictorially above. Its many parts are listed in detail below in the

order in which they should occur in the system deck. An OP. to the right of a given entry means optional. Optional portions of the system deck are included only when the particular feature that they provide is required.

1. Run Request Form
2. \$JOB Card
3. System Control Cards

\$SETUP UT8	P5330, NORING	
\$ASSIGN	SYSUT8	
\$ASSIGN	SYSUT9	
\$ASSIGN	SYSUT7	
\$ASSIGN	SYSUT6	
\$SETUP PL1	DISK, PLOT,,,1	OP.
\$ASSIGN	SYSPL1	OP.
\$SETUP UTO	DISK,PUNCH	OP.
\$ASSIGN	SYSUTO	OP.
\$ATTEND	0,77777	OP.
4. Fortran Monitor Control Cards

*	XEQ
*	LØAD 12
5. TAG Description Deck
6. All Fortran Subroutine Decks OP.
7. All Fap subroutine Decks OP.
8. All Binary Decks

Component Modeling Subroutines	OP.
Premanent Execution System Subroutines	
9. Fortran Monitor Control Card * DATA
10. Data Deck
11. End of File Card EOF

2. TAG Preprocessor Run

This set up is used whenever the solution program itself is the only desired output of a TAG

run. The contents required in such a TAG system deck are listed below in the order in which they should be placed.

1. Run Request Form
2. \$JOB Card
3. System Control Cards

\$SETUP UT8	P5530,NORING	
\$ASSIGN	SYSUT8	
\$ASSIGN	SYSUT9	
\$ASSIGN	SYSUT7	
\$ASSIGN	SYSUT6	
\$ATTEND	0,77777	OP.
4. Fortran Monitor Control Card * LØAD 12
5. TAG Description Deck.
6. End of File Card EOF

3. Fortran Simulation Program Run

This setup is used whenever it is desired to run a circuit simulation using the Fortran program output of a previous TAG preprocessor run. The contents required in such a TAG system deck are listed below.

1. Run Request Form
2. \$JOB Card
3. System Control Cards

\$SETUP PL1	DISK,PLØT,,,1	OP.
\$ASSIGN	SYSPL1	OP.
\$SETUP UT0	DISK,PUNCH	OP.
\$ASSIGN	SYSUT0	OP.
\$ATTEND	0,77777	OP.
4. Fortran Monitor Control Card * XEQ
5. Fortran Simulation Program
6. All Fortran Subroutine Decks OP.

7. All FAP Subroutine Decks OP.
8. All Binary Decks
Component Modeling Subroutines OP.
Permanent Executions System Subroutines.
9. Fortran Monitor Control Card * DATA
10. Data Deck
11. End of File Card EOF

4. Binary Simulation Program Run

One of the system outputs of every Fortran simulation program run is the Fortran compiled binary equivalent of that program in card form. This relocatable machine language deck may be used in the TAG system deck for further simulation runs in the same way that the Fortran simulation program is used. The cards and decks required in such a TAG system deck are listed below in the order in which they should be placed.

1. Run Request Form
2. \$JOB Card
3. System Control Cards

\$SETUP PL1	DISK, PL0T, , , 1	OP.
\$ASSIGN	SYSPL1	OP.
\$SETUP UT0	DISK, PUNCH	OP.
\$ASSIGN	SYSUT0	OP.
\$ATTEND	0,77777	OP.
4. Fortran Monitor Control Card * XEQ
5. All Fortran Subroutine Decks OP.
6. All FAP Subroutine Decks OP.
7. All Binary Decks
Binary Simulation program
Component Modeling Subroutines OP.
Permanent Execution System Subroutines

- 8. Fortran Monitor Control Card * DATA
- 9. Data Deck
- 10. End of File Card EOF

5.0 EXAMPLE TAG ANALYSES

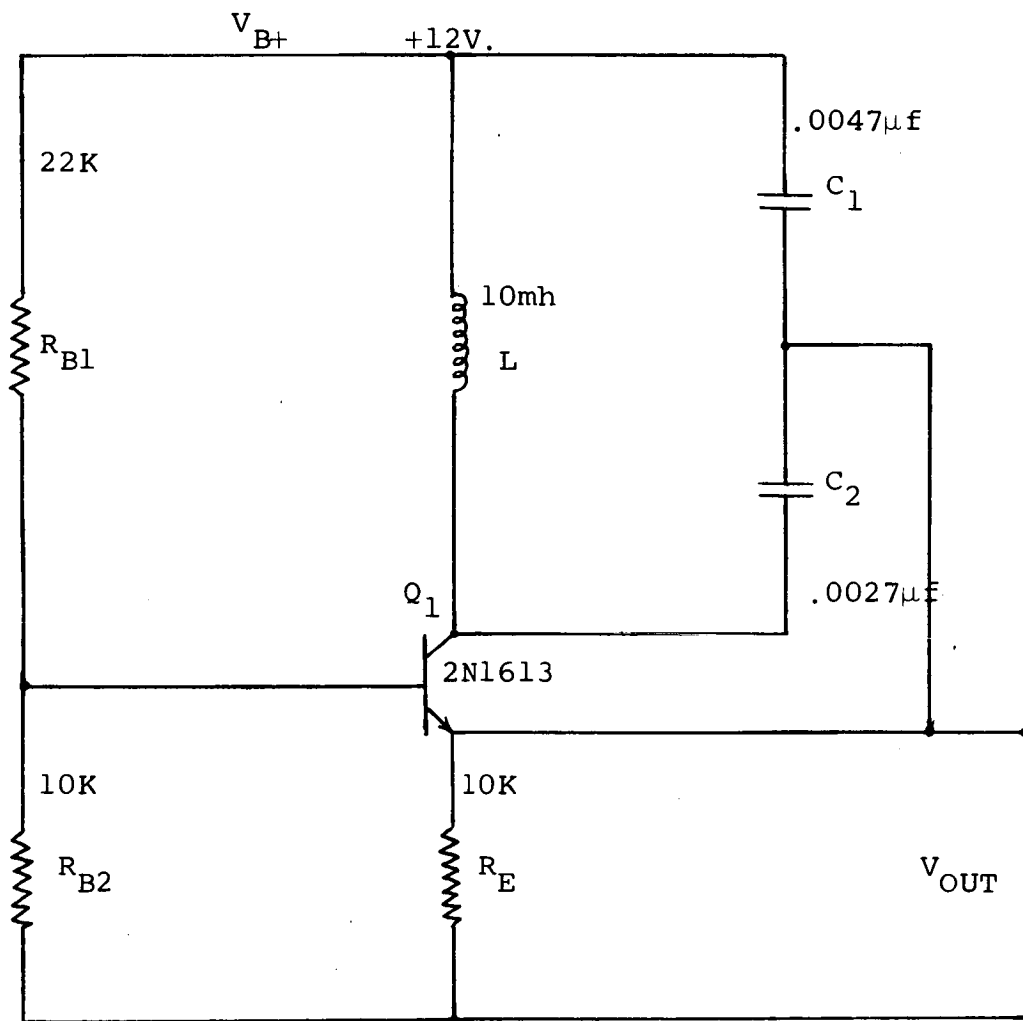
A. Introduction

Two example analyses are provided in this section as an aid to the beginner and occasional user. The first example is a simple single transistor colpitts oscillator circuit using the non-linear Ebers-Moll transistor model to characterize the behavior of the transistor. This first example is set up in great detail as a rather idealized example. The whole output listing is provided in this example as well as a complete list of output plots. The second example circuit is a simple saturating inductor inverter or flux oscillator. This example is presented in much less detail but does demonstrate the use of linear segmented modeling in transient simulation. The Ebers-Moll transistor model and the three piece linear segmented saturating inductor model subroutines which were used in these two examples are available for for reference at the end of section 3.C.

B. Example Colpitts Oscillator Simulation

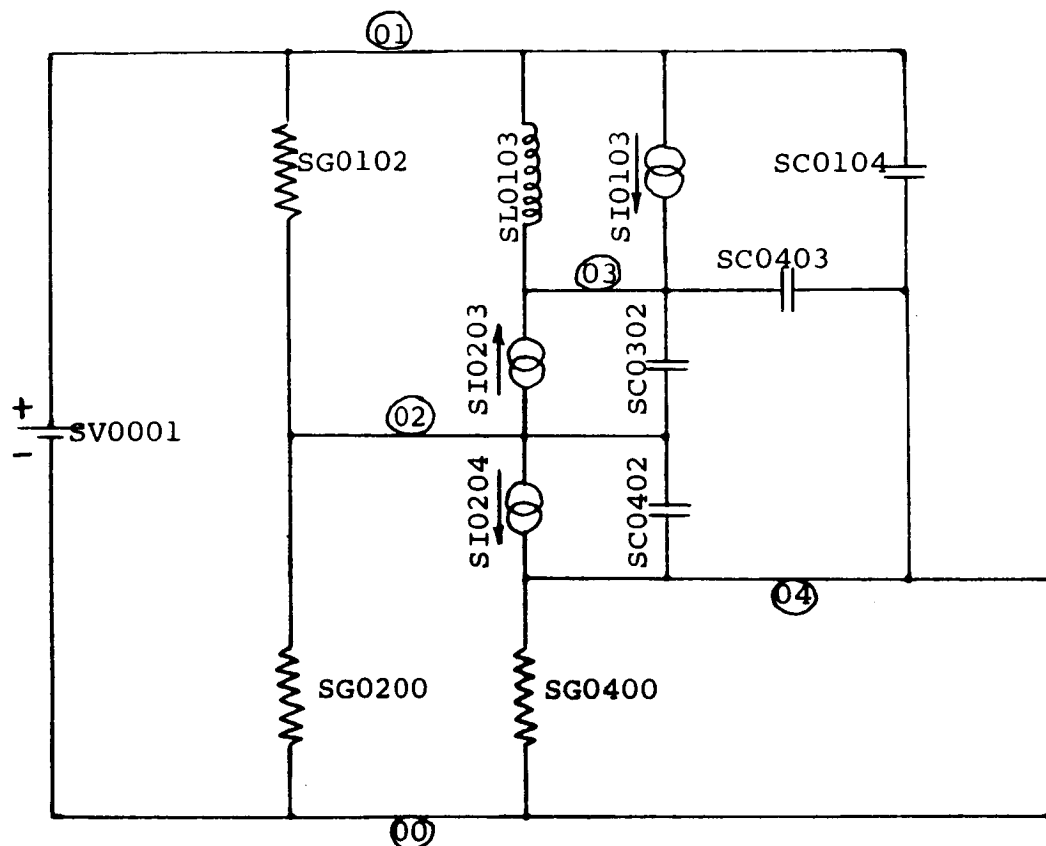
1. Problem Preparation

All typical steps required in setting up the problem are shown below in detail except for measuring the transistor characteristics and calculating the parameters required by the model used. The process starts with the circuit schematic as shown below and the set of circuit parameter values for which the simulation is to be performed.



Colpitts Oscillator
Original Circuit Schematic

2. Construction of TAG Equivalent Circuit



This step demonstrates two of the inconveniences of the present TAG system which could easily be remedied. The first is that the TAG input language, while easy to master, is not as close to that used by the engineer as could be achieved. The second is that, since TAG recognizes only voltage sources, capacitances, conductances, reciprocal inductances, current sources, and ideal transformer turns per winding as components, non-standard components such as transistors or inductor initial conditions must be transformed into an equivalent standard set of components. It is often helpful to devise a transformation table

as shown below which gives all the information required by TAG and relates it to the original circuit schematic.

Users Ident.	TAG Ident.				Value in TAG units	Function
	S	Type	Node Nos.			
			-	+		
V _{B+}	S	V	00	01	12. Volts	B+ fixed voltage supply
C ₁	S	C	01	04	4.7E-9 farads	Part of Feedback and Tank
C ₂	S	C	04	03	2.7E-9 farads	Part of Feedback and Tank
L	S	L	01	03	100. henrys ⁻¹	Part of tank and collec- tor load
	S	I	01	03	0.0 amps	Initial value of induc- tor current
R _E	S	G	04	00	1.00E-4 mhos	Emitter bias resistor
R _{B1}	S	G	01	02	4.55E-5 mhos	One of two base bias resistors
R _{B2}	S	G	02	00	1.00E-4 mhos	Second of two base bias resistors
C _{EQ1}	S	C	04	02	[Calculated by the TRAN3 subroutine for Q ₁]	Emitter junction capaci- tance
C _{CQ1}	S	C	03	02		Collector junction capa- citance
I _{EQ1}	S	I	02	04		Total emitter dependent current
I _{CQ1}	S	I	02	03		Total collector depen- dent current
	S	V	01	04		-8.85 volts
	S	V	04	03	3.00 volts	C ₂ initial voltage value

3. Creation of Coding Sheet Description of TAG

Simulation - The branch list may be copied onto the coding sheets either directly from the equivalent circuit topology or from the transformation table. This is shown on the first coding sheet below.

Next, the parameter description statements for all non-standard components are listed. This is shown on the second coding sheet where the transistor subroutine call statement, CALL TRAN3, is listed. Its arguments refer to the equivalent TAG components which are shown in place of the transistor in the TAG equivalent circuit. The subroutine relates the values of collector and emitter current source currents and capacitances to the transistor junction voltages. A listing of this subroutine in Fortran II source language form is given at the end of the simulation listing.

The second coding sheet also contains a DIMENSION statement which sets aside extra computer storage space for the multi-valued variables named. (See a Fortran manual for explanation.)

The third coding sheet demonstrates the ease of designating and labeling variables to be provided as outputs during the simulation. Each of the circuit variables named here will be printed out in a column at a interval determined by the time variable FOUT and labeled according to the set of characters at the left of the equal sign. Placing the print control character, D (for double space) and S (for single space), in column 73 has led to

some confusion on the part of the keypunch operators since this represents somewhat of a non-standard operation.

The fourth coding sheet demonstrates one way to use the automatic plot routine provided with the TAG system. There is unfortunately much unnecessary and redundant coding required by this method. The redundancy can be reduced somewhat by changing the coding slightly, however, it remains a somewhat less than optimum situation.

Simulation termination control is also provided on this sheet by the IF, GO TO, and END statements. Transient simulation is continued until FT, simulated time, becomes equal to or larger than DONE, termination time. At that time the second set of plot statements are entered and the scaling is finished and the axis properly labeled. All computational matrices are then zeroed and the program looks for another set of parameter data. If no more input data exists the program terminates.

The final coding sheet demonstrates the input format required by parameter data. Most of the entries are single valued and self-explanatory. However, the DATA entry can be seen to have 17 values. DATA is entered as an ordered array of the 17 parameters required by the transistor subroutine, TRAN3. The function of each is given in order with the TRAN3 source deck listing.

Program <u>Colpitt's Oscillator - Topology</u> Date _____ Programmer _____						Punching Instructions	Page <u>1</u> of <u>5</u>
							*
						Card Form #	Identification
						Graphic	[' + . - *]
						Punch	[73 + - *] 80

STATEMENT NUMBER		FORTRAN STATEMENT
	6	SV0001,
	7	SC0104, SC0403, SC0402, SC0302,
	8	SG0102, SG0200, SG0400,
	9	SL0103,
	10	SI0204, SI0203, SI0103*
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	
	50	
	51	
	52	
	53	
	54	
	55	
	56	
	57	
	58	
	59	
	60	
	61	
	62	
	63	
	64	
	65	
	66	
	67	
	68	
	69	
	70	
	71	
	72	



FORTRAN CODING FORM

	Punching Instructions	Page 3 of 5
Program Colpitts Oscillator - Print Outputs Programmer _____	Graphic	* Card Form #
Date _____	Punch	Identification 73 _____ 80

-- C FOR COMMENT

[illegible]

* A standard card form, IBM, electro 888157, is available for punching source statements from this form.

FORTRAN CODING FORM

Program <u>Colpitts Oscillator - Plotted Outputs</u>		Punching Instructions		Page <u>4</u> of <u>5</u>
Programmer <u>and Termination Control</u>	Date	Graphic	Card Form #	Identification
		Punch		

--- C FOR COMMENT

STATEMENT NUMBER	FORTRAN STATEMENT	73	74	75	76	77	78	79	80
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40									
41									
42									
43									
44									
45									
46									
47									
48									
49									
50									
51									
52									
53									
54									
55									
56									
57									
58									
59									
60									
61									
62									
63									
64									
65									
66									
67									
68									
69									
70									
71									
72									
73									
74									
75									
76									
77									
78									
79									
80									

These five sheets represent the total description supplied by the user to the TAG analysis program. From this, TAG will automatically perform a single simulation run on the Colpitts oscillator circuit. The cards generated from these sheets are then combined with the computing system control cards and the TAG execution deck and submitted to the computing system operator.

C. Simulation Output

What follows are the listings returned to the TAG user by the machine operator. Included is a listing of the users inputs, a listing the TAG generated solution program, and a list of data outputs which trace a time history of the circuit under analysis. At the end of the listings is a set of curves which show several of the output variables plotted against time.

COMPUTER LISTING
TAG CIRCUIT DESCRIPTION

\$JOB CIB,550200C,40007-0,73877,A FC1,10 G. A. PACKER A

90 UNIT	RD	PU	PR	A1	A2	A3	A4	A5	A6	A7	A8	A9
FUNCTION	CRD	PCH	PRT	LBI	INI	QUI	PPI	CKI	UT8	PL1	UTC	
SYMBOLIC												
40 LOGICAL	32	33	34	00	01	02	03	04	05	06	07	08
40 UNIT	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK

90 UNIT	B8	B9	B0	C1	C2	C3	C4	C5	C6	D1	D2	D3
FUNCTION												
SYMBOLIC												
40 LOGICAL	17	18	19	20	21	22	23	24	25	26	27	28
40 UNIT	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK	DISK

9 095122	0	\$SETUP	UT8	P5330,NGRING
9 095122	0	\$ASSIGN		SYSUT8
9 095122	0	\$ASSIGN		SYSUT9
9 095122	0	\$ASSIGN		SYSUT7
9 095122	0	\$ASSIGN		SYSUT6
9 095122	0	\$SETUP	PL1	DISK,PLOT,,,1
9 095122	0	\$ASSIGN		SYSPL1
9 095122	0	\$SETUP	UTO	DISK,PUNCH
9 095122	0	\$ASSIGN		SYSUTO
9 095122	0	\$ATEND		0,77777

System Control
Tape Unit Setup Cards
Memory Dump Control

XEQ
LOAD(12)

Fortran Execution Card
Load TAG Program From Tape Unit UT8

CIRCUIT DESCRIPTION

SV0001,

SC0104, SC0403, SC0402, SC0302,

SG0102, SG0200, SG0400,

SL0103,

SI0204, SI0203, SI0103*

	NUMBER OF NODES IS	4
2SC 55636 MAX	340 COUNT	18

TRANSIENT ANALYSIS GENERATOR (W.J.THOMAS-JP

```

CALL TRAN3(SV0402, SV0302, $SI0204, $SI0203, $SC0402, $SC0302, DATA,
1CIE1, CIE1, LALGFT, KTDC)
DIMENSION DATA(17), BF1(100), BF2(100), BF3(100), BF4(100), BF5(100)
TIME = FT
VIN = SV0001
VOUT= SV0004
VCLTR= SV0003
VBASE= SV0002
VRE = SV0402
VCB = SV0203
CURL = $S0301*$S0103
CCLTR=-$SI0203
CENTR= $SI0204
CBASE = $SI0203+$SI0204
CALL SCOPE(FT, VOUT, BF1, 100, 1, 1H1, 1H1)
CALL SCOPE(FT, VCB, BF2, 100, 1, 1H1, 1H1)
CALL SCOPE(FT, VRE, BF3, 100, 1, 1H1, 1H1)
CALL SCOPE(FT, CCLTR, BF4, 100, 1, 1H1, 1H1)
CALL SCOPE(FT, CBASE, BF5, 100, 1, 1H1, 1H1)
IF(FT-DONE) 6200, 60, 60
60 CALL SCOPE(FT, VOUT, BF1, 100, 2, 10HVOUT VOLTS, 9HTIME SECS)
CALL SCOPE(FT, VCB, BF2, 100, 2, 9HVCE VOLTS, 9HTIME SECS)
CALL SCOPE(FT, VRE, BF3, 100, 2, 9HVRE VOLTS, 9HTIME SECS)
CALL SCOPE(FT, CCLTR, BF4, 100, 2, 7HIC AMPS, 9HTIME SECS)
CALL SCOPE(FT, CBASE, BF5, 100, 2, 10HIBASE AMPS, 9HTIME SECS)
GO TO 6000
END

```

C
S
S
S
S
S
S
S
S
S
S

COMPUTER LISTING

TAG GENERATED

SOLUTION PROGRAM

```

DIMENSION FS21(3),FV21(3),FC12(1,3),FG12(1,3),FL12(1,3),FVD21(3),F
1C22(3,3),FI21(3),FG21(3),FG22(3,3),FL21(3),FL22(3,3),FSD21(3),FC21
2(3),FC122(3,3)
DIMENSION FHB(6),LNH(6),FMIS(182)
COMMON FMIS,FVD21,FSD21,FV21,FS11,FS21,FI2,F1,LNF,FHP
DIMENSION FCMDMY(2790)
COMMON FCMDMY
DIMENSION FTEM1(3),FTEM2(3)
LNV=1
LNC=3
LNG=0
LNL=0
FSTEP=1.E-11
FEPSL2=5.E-6
FEPSL3=5.E-4
FEPSL4=1.E-16
LTYPE=4
GO TO 6000
DIMENSION DATA(17),BF1(100),BF2(100),BF3(100),BF4(100),BF5(100)
30 TIME=FT
WRITE OUTPUT TAPE 6,8000,TIME
8000 FORMAT (1H0,45X,5HTIME=,E16.8)
VIN=+FV11
WRITE OUTPUT TAPE 6,8001,VIN
8001 FORMAT (1H ,46X,4HVIN=,E16.8)
VOUT=+FV11+FV21(1)
WRITE OUTPUT TAPE 6,8002,VOUT
8002 FORMAT (1H ,45X,5HVOUT=,E16.8)
VCLTR=+FV11+FV21(1)+FV21(2)
WRITE OUTPUT TAPE 6,8003,VCLTR
8003 FORMAT (1H ,44X,6HVCLTR=,E16.8)
VBASE=+FV11+FV21(1)+FV21(3)
WRITE OUTPUT TAPE 6,8004,VBASE
8004 FORMAT (1H ,44X,6HVBASE=,E16.8)
VBE=+FV21(3)
WRITE OUTPUT TAPE 6,8005,VBE
8005 FORMAT (1H ,46X,4HVBE=,E16.8)
VCE=+FV21(2)-FV21(3)
WRITE OUTPUT TAPE 6,8006,VCE
8006 FORMAT (1H ,46X,4HVCE=,E16.8)
CURL=(-FS21(1)-FS21(2))*SL01C3
WRITE OUTPUT TAPE 6,8007,CURL
8007 FORMAT (1H ,45X,5HCURL=,E16.8)
CCLTR=-SI02C3
WRITE OUTPUT TAPE 6,8008,CCLTR
8008 FORMAT (1H ,44X,6HCCLTR=,E16.8)
CENTR=SI02C4
WRITE OUTPUT TAPE 6,8009,CENTR
8009 FORMAT (1H ,44X,6HCENTR=,E16.8)
CBASE=SI02C3+SI02C4
WRITE OUTPUT TAPE 6,8010,CBASE
8010 FORMAT (1H ,44X,6HCBASE=,E16.8)
CALL SCOPE(FT,VOUT,BF1,100,1,1H1,1H1)
CALL SCOPE(FT,VCE,BF2,100,1,1H1,1H1)
CALL SCOPE(FT,VBE,BF3,100,1,1H1,1H1)
CALL SCOPE(FT,CCLTR,BF4,100,1,1H1,1H1)

```

```

CALL SCOPE(FT,CBASE,BF5,100,1,1H1,1H1)
IF(FT-DONE)6200,60,60
CALL SCOPE(FT,VOUT,BF1,100,2,10HVOUT VOLTS,9HTIME SECS)
CALL SCOPE(FT,VCE,BF2,100,2,9HVCE VOLTS,9HTIME SECS)
CALL SCOPE(FT,VBE,BF3,100,2,9HVBE VOLTS,9HTIME SECS)
CALL SCOPE(FT,CCLTR,BF4,100,2,7HIC AMPS,9HTIME SECS)
CALL SCOPE(FT,CBASE,BF5,100,2,10HIBASE AMPS,9HTIME SECS)
GO TO 6000

```

```

6000 CALL ZEROX(FS11,1)
CALL ZEROX(FS21,3)
CALL ZEROX(FV11,1)
CALL ZEROX(FV21,3)
CALL ZEROX(FC11,1)
CALL ZEROX(FI11,1)
CALL ZEROX(FC12,3)
CALL ZEROX(FG11,1)
CALL ZEROX(FG12,3)
CALL ZEROX(FL11,1)
CALL ZEROX(FL12,3)
CALL ZEROX(FVD21,3)
CALL ZEROX(FC22,9)
CALL ZEROX(FI21,3)
CALL ZEROX(FG21,3)
CALL ZEROX(FG22,9)
CALL ZEROX(FL21,3)
CALL ZEROX(FL22,9)
CALL ZEROX(FSD11,1)
CALL ZEROX(FSD21,3)
CALL ZEROX(FC21,3)
CALL ZEROX(FVD11,1)
CALL ZEROX(FV011,1)
FT=0.

```

```

CALL INPUT (6HSI0103,SI0103,6HSI0203,SI0203,6HSI0204,SI0204,6HSL01
103,SL0103,6HSG0400,SG0400,6HSG0200,SG0200,6HSG0102,SG0102,6HSC0302
2,SC0302,6HSC0402,SC0402,6HSC0403,SC0403,6HSC0104,SC0104,6HSV0302,S
3V0302,6HSV0402,SV0402,6HSV0403,SV0403,6HSV0104,SV0104,6HSV0001,SV0
4001,2HFT,FT,5HFSTEP,FSTEP,5HFEP SL,FEPSL,6HFEP SL1,FEPSL1,6HFEP SL2,F
5EP SL2,6HFEP SL3,FEPSL3,6HFEP SL4,FEPSL4,4HFOUT,FOUT,6HLDBG01,LDBG01,
64HLMAX,LMAX,5HLTYPE,LTYPE,6HLALGFT,LALGFT,4HDATA,DATA,4HDONE,DONE,
74HCIE1,CIE1,4HCIC1,CIC1,5HCCLTR,CCLTR,5HCBASE,CBASE,4HKTDC,KTDC,3H
8BF1,BF1,3HBF2,BF2,3HBF3,BF3,3HBF4,BF4,3HBF5,BF5,4HVOUT,VOUT,3HVCE,
9VCE)

```

```

CALL INPUT (3HVBE,VBE,6H$$$END)

```

```

6100 CONTINUE
LALGFT=1
FHB(1)=FSTEP
FHB(2)=1.E-5
FHB(3)=FEPSL4
FHB(4)=.5
FHB(5)=FEPSL2
FHB(6)=FEPSL3
LNH(1)=7
LNH(2)=LNH(1)
LNH(5)=5
LINT=0
LCNT=1

```

FT2=0.

FT0=FT

FV11=+SV0001

FV21(1)=+SV0104

FV21(2)=+SV0403

FV21(3)=+SV0402

FC22(1,1)=+SC0104

FG11=+SG0200+SG0400

FG12(1,1)=+SG0200+SG0400

FG12(1,3)=+SG0200

FG21(1)=+SG0200+SG0400

FG22(1,1)=+SG0102+SG0200+SG0400

FG22(1,3)=+SG0102+SG0200

FG21(3)=+SG0200

FG22(3,1)=+SG0102+SG0200

FG22(3,3)=+SG0102+SG0200

FL22(1,1)=+SL0103

FL22(1,2)=+SL0103

FL22(2,1)=+SL0103

FL22(2,2)=+SL0103

FI21(1)=+SI0103

CALL RSTOP (FSTOP,FT,FSTEP)

6200 IF(LCNT-3)6202,6201,6201

6201 CALL STOP(FOUT,LINT)

6202 CONTINUE

GO TO (6395,6390),LALGFT

6390 CALL ROUT (LINT)

6395 CALL FMARK(LCNT,FHB,0,LTYPE,1,1,0,FHC,3,FT,FSTCP,0)

IF(LCNT-2)6300,6300,30

6300 CONTINUE

CALL TRAN3((+FV21(3)),(-FV21(2)+FV21(3)),SI0204,SI0203,SC0402,SC03

102,DATA,CIE1,CIC1,LALGFT,KTDC)

FC22(2,2)=+SC0403+SC0302

FC22(2,3)=-SC0302

FC22(3,2)=-SC0302

FC22(3,3)=+SC0402+SC0302

FI21(2)=+SI0203+SI0103

FI21(3)=-SI0204-SI0203

LALGFT=2

FDLT=FT-FT0

IF(FDLT)7030,7030,7025

7025 FVD11=(FV11-FV011)/FDLT

7030 FV011=FV11

FT0=FT

CALL INV(FC22,FCI22,LNC)

CFVD21=FCI22*(FI21-FG21*FV11-FG22*FV21-FL21*FS11-FL22*FS21-FC21*FVD11)

CALL MULT(FC21,FVD11,FTEM1,3,1,1)

CALL MULT(FL22,FS21,FTEM2,3,1,3)

CALL PSUM(FTEM2,FTEM1,FTEM1,3,1)

CALL MULT(FL21,FS11,FTEM2,3,1,1)

CALL PSUM(FTEM2,FTEM1,FTEM1,3,1)

CALL MULT(FG22,FV21,FTEM2,3,1,3)

CALL PSUM(FTEM2,FTEM1,FTEM1,3,1)

CALL MULT(FG21,FV11,FTEM2,3,1,1)

CALL PSUM(FTEM2,FTEM1,FTEM1,3,1)

CALL MSUM(FI21,FTEM1,FTEM1,3,1)

TRANSIENT ANALYSIS GENERATOR (W.J.THOMAS-JPI)

CALL MULT(FC122,FTEM1,FVD21,3,1,3)

FSD11=FV11

DO 7042 L=1,LNC

7042 FSD21(L)=FV21(L)

CALL ROUT(0)

END(1,0,0,0,0,0,1,0,0,0,0,0,0,0,0)

2SC 41305 MAX 1413 COUNT 466

COMPUTER LISTING
SAMPLE TRANSIENT SIMULATION

(UNIT)	43426		
ENDFIL	43451		
(IOU)	44063	00000	44060
PDUMP	44115	00003	44111
DUMP	44114		
XMIN	44340	00000	44340
MINUTE	44340		
CLOCK	44350		

UNUSED CORE 44404 THRU 71571

EXECUTION 115237

SV0001= 12.0,
 SC0104= 4.7E-9, SC0403= 2.7E-9,
 SG0102= 4.55E-5, SG0200= 1.E-4, SG0400= 1.E-4,
 SL0103 = 100.,
 DATA= .75E-13, 1.49E-13, 1.E-9, 1.E-9, .991, .5, .199E-8, .2E-7, .75, .75,
 .50, .33, 95.E-12, 33.E-12, 0.0, 0.0, 0.026 ,
 SI0103= 0.0, SV0104=-8.85, SV0403=3.0,
 KTDC =0, CIE1=0., CIC1=0.,
 FOUT=.5E-6, DONE=1.E-4*

Note $V_{CE} = V_{CB}$

TIME= 0.
 VIN= 0.12000000E 02
 VOUT= 0.31500001E 01
 VCLTR= 0.61500000E 01
 VRASE= 0.31500001E 01
 VBE= 0.
 VCE= 0.30000000E 01
 CURL= -0.
 CCLTR= 0.30002946E-08
 CEMTR= 0.14732408E-12
 CBASE= -0.30001473E-08

TIME= 0.49999999E-06
 VIN= 0.12000000E 02
 VOUT= 0.31394648E 01
 VCLTR= 0.61682902E 01
 VBASE= 0.33863511E 01
 VBE= 0.24688631E-00
 VCE= 0.27819391E 01
 CURL= 0.29237662E-03
 CCLTR= 0.47417086E-08
 CEMTR= 0.22243039E-08
 CBASE= -0.25174046E-08

TIME= 0.09999999E-05
 VIN= 0.12000000E 02
 VOUT= 0.31569614E 01
 VCLTR= 0.62672298E 01
 VRASE= 0.35394736E 01
 VBE= 0.38251217E-00
 VCE= 0.27277561E 01
 CURL= 0.58182511E-03
 CCLTR= 0.36381918E-06
 CEMTR= 0.36475313E-06
 CBASE= 0.93394447E-09

TIME= 0.15000000E-05
 VIN= 0.12000000E 02
 VOUT= 0.32025065E 01
 VCLTR= 0.64458998E 01

VBASE= 0.36592404E 01
VBE= 0.45673393E-00
VCE= 0.27866593E 01
CURL= 0.86432275E-03
CCLTR= 0.62745571E-05
CEMTR= 0.63291856E-05
CBASE= 0.54628458E-07

TIME= 0.20000000E-05
VIN= 0.12000000E 02
VOUT= 0.32751995E 01
VCLTR= 0.67001860E 01
VBASE= 0.37651154E 01
VBE= 0.48991597E-00
VCE= 0.29350705E 01
CURL= 0.11359748E-02
CCLTR= 0.22475403E-04
CEMTR= 0.22677046E-04
CBASE= 0.20164384E-06

TIME= 0.24999999E-05
VIN= 0.12000000E 02
VOUT= 0.33735574E 01
VCLTR= 0.70265195E 01
VBASE= 0.38643297E 01
VBE= 0.49077230E-00
VCE= 0.31621898E 01
CURL= 0.13930994E-02
CCLTR= 0.23228103E-04
CEMTR= 0.23436355E-04
CBASE= 0.20825132E-06

TIME= 0.30000000E-05
VIN= 0.12000000E 02
VOUT= 0.34956723E 01
VCLTR= 0.74239041E 01
VBASE= 0.39574259E 01
VBE= 0.46175362E-00
VCE= 0.34664781E 01
CURL= 0.16321397E-02
CCLTR= 0.76108846E-05
CEMTR= 0.76769684E-05
CBASE= 0.66083714E-07

TIME= 0.34999999E-05
VIN= 0.12000000E 02
VOUT= 0.36393169E 01
VCLTR= 0.78868257E 01
VBASE= 0.40399669E 01
VBE= 0.40065013E-00
VCE= 0.38468587E 01
CURL= 0.18496244E-02
CCLTR= 0.72926261E-06
CEMTR= 0.73240428E-06
CBASE= 0.31416789E-08

TIME= 0.40000000E-05
VIN= 0.12000000E 02
VOUT= 0.38020223E 01
VCLTR= 0.84069625E 01
VBASE= 0.41064414E 01
VBE= 0.30441920E-00
VCE= 0.43005209E 01
CURL= 0.20424963E-02
CCLTR= 0.22214178E-07
CEMTR= 0.18380614E-07

CBASE= -0.38335643E-08

TIME= 0.45000000E-05
VIN= 0.12000000E 02
VOUT= 0.39811186E 01
VCLTR= 0.89763983E 01
VBASE= 0.41531225E 01
VBE= 0.17200390E-00
VCE= 0.48232757E 01
CURL= 0.22081030E-02
CCLTR= 0.49334158E-08
CEMTR= 0.28299424E-09
CBASE= -0.46504215E-08

TIME= 0.49999999E-05
VIN= 0.12000000E 02
VOUT= 0.41737361E 01
VCLTR= 0.95870087E 01
VBASE= 0.41795181E 01
VBE= 0.57819923E-02
VCE= 0.54074906E 01
CURL= 0.23441719E-02
CCLTR= 0.54076219E-08
CEMTR= 0.59663408E-11
CBASE= -0.54018555E-08

TIME= 0.55000000E-05
VIN= 0.12000000E 02
VOUT= 0.43768000E 01
VCLTR= 0.10230179E 02
VBASE= 0.41862250E 01
VBE= -0.18857501E-00
VCE= 0.60419544E 01
CURL= 0.24488591E-02
CCLTR= 0.60421018E-08
CEMTR= -0.18857624E-09
CBASE= -0.62306780E-08

TIME= 0.59999999E-05
VIN= 0.12000000E 02
VOUT= 0.45870411E 01
VCLTR= 0.10896792E 02
VBASE= 0.41834828E 01
VBE= -0.40355828E-00
VCE= 0.67133088E 01
CURL= 0.25207629E-02
CCLTR= 0.67134561E-08
CEMTR= -0.40355962E-09
CBASE= -0.71170156E-08

TIME= 0.64999999E-05
VIN= 0.12000000E 02
VOUT= 0.48010529E 01
VCLTR= 0.11577369E 02
VBASE= 0.41704301E 01
VBE= -0.63062280E 00
VCE= 0.74069398E 01
CURL= 0.25589366E-02
CCLTR= 0.74070871E-08
CEMTR= -0.63062412E-09
CBASE= -0.80377112E-08

TIME= 0.70000000E-05
VIN= 0.12000000E 02
VOUT= 0.50152864E 01
VCLTR= 0.12262134E 02

VBASE= 0.41514263E 01
VBE= -0.86386004E 00
VCE= 0.81107079E 01
CURL= 0.25629473E-02
CCLTR= 0.81108551E-08
CEMTR= -0.86386137E-09
CBASE= -0.89747164E-08

TIME= 0.74999999E-05
VIN= 0.12000000E 02
VOUT= 0.52261700E 01
VCLTR= 0.12941238E 02
VBASE= 0.41272534E 01
VBE= -0.10989166E 01
VCE= 0.88139851E 01
CURL= 0.25328192E-02
CCLTR= 0.88141324E-08
CEMTR= -0.10989179E-08
CBASE= -0.99130503E-08

TIME= 0.80000000E-05
VIN= 0.12000000E 02
VOUT= 0.54302147E 01
VCLTR= 0.13604970E 02
VBASE= 0.40998241E 01
VBE= -0.13303905E 01
VCE= 0.95051456E 01
CURL= 0.24690771E-02
CCLTR= 0.95052928E-08
CEMTR= -0.13303918E-08
CBASE= -0.10835684E-07

TIME= 0.84999999E-05
VIN= 0.12000000E 02
VOUT= 0.56240498E 01
VCLTR= 0.14243869E 02
VBASE= 0.40717696E 01
VBE= -0.15522800E 01
VCE= 0.10172100E 02
CURL= 0.23727322E-02
CCLTR= 0.10172247E-07
CEMTR= -0.15522814E-08
CBASE= -0.11724529E-07

TIME= 0.89999998E-05
VIN= 0.12000000E 02
VOUT= 0.58044732E 01
VCLTR= 0.14848880E 02
VBASE= 0.40461891E 01
VBE= -0.17582840E 01
VCE= 0.10802690E 02
CURL= 0.22452579E-02
CCLTR= 0.10802837E-07
CEMTR= -0.17582852E-08
CBASE= -0.12561122E-07

TIME= 0.94999997E-05
VIN= 0.12000000E 02
VOUT= 0.59683210E 01
VCLTR= 0.15411230E 02
VBASE= 0.40150169E 01
VBE= -0.19533041E 01
VCE= 0.11396214E 02
CURL= 0.20885620E-02
CCLTR= 0.11396360E-07
CEMTR= -0.19533055E-08

CBASE= -0.1334960E-07

TIME= 0.9999997E-05
VIN= 0.12000000E 02
VOUT= 0.61128665E 01
VCLTR= 0.15923038E 02
VBASE= 0.39814866E 01
VBE= -0.21313798E 01
VCE= 0.11941551E 02
CURL= 0.19049776E-02
CCLTR= 0.11941699E-07
CEMTR= -0.21313812E-08
CBASE= -0.14073080E-07

TIME= 0.10500000E-04
VIN= 0.12000000E 02
VOUT= 0.62355813E 01
VCLTR= 0.16377050E 02
VBASE= 0.39424843E 01
VBE= -0.22930970E 01
VCE= 0.12434566E 02
CURL= 0.16972119E-02
CCLTR= 0.12434713E-07
CEMTR= -0.22930983E-08
CBASE= -0.14727811E-07

TIME= 0.10999999E-04
VIN= 0.12000000E 02
VOUT= 0.63345920E 01
VCLTR= 0.16767310E 02
VBASE= 0.39178058E 01
VBE= -0.24167861E 01
VCE= 0.12849504E 02
CURL= 0.14683302E-02
CCLTR= 0.12849651E-07
CEMTR= -0.24167874E-08
CBASE= -0.15266438E-07

TIME= 0.11499999E-04
VIN= 0.12000000E 02
VOUT= 0.64077862E 01
VCLTR= 0.17088006E 02
VBASE= 0.38856459E 01
VBE= -0.25221402E 01
VCE= 0.13202360E 02
CURL= 0.12216530E-02
CCLTR= 0.13202507E-07
CEMTR= -0.25221415E-08
CBASE= -0.15724649E-07

TIME= 0.11999999E-04
VIN= 0.12000000E 02
VOUT= 0.64537333E 01
VCLTR= 0.17334748E 02
VBASE= 0.38446183E 01
VBE= -0.26091150E 01
VCE= 0.13490130E 02
CURL= 0.96076491E-03
CCLTR= 0.13490277E-07
CEMTR= -0.26091163E-08
CBASE= -0.16099393E-07

TIME= 0.12499999E-04
VIN= 0.12000000E 02
VOUT= 0.64716871E 01
VCLTR= 0.17504610E 02

VBASE= 0.38138505E 01
VBE= -0.26578366E 01
VCE= 0.13690759E 02
CURL= 0.68945828E-03
CCLTR= 0.13690907E-07
CEMTR= -0.26578379E-08
CBASE= -0.16348745E-07

TIME= 0.12999999E-04
VIN= 0.12000000E 02
VOUT= 0.64607625E 01
VCLTR= 0.17595039E 02
VBASE= 0.37724585E 01
VBE= -0.26883039E 01
VCE= 0.13822581E 02
CURL= 0.41162857E-03
CCLTR= 0.13822728E-07
CEMTR= -0.26883052E-08
CBASE= -0.16511033E-07

TIME= 0.13499999E-04
VIN= 0.12000000E 02
VOUT= 0.64210972E 01
VCLTR= 0.17605449E 02
VBASE= 0.37432732E 01
VBE= -0.26778240E 01
VCE= 0.13862175E 02
CURL= 0.13128620E-03
CCLTR= 0.13862323E-07
CEMTR= -0.26778253E-08
CBASE= -0.16540148E-07

TIME= 0.13999999E-04
VIN= 0.12000000E 02
VOUT= 0.63525980E 01
VCLTR= 0.17535527E 02
VBASE= 0.37012579E 01
VBE= -0.26513401E 01
VCE= 0.13834269E 02
CURL= -0.14757606E-03
CCLTR= 0.13834416E-07
CEMTR= -0.26513415E-08
CBASE= -0.16485757E-07

TIME= 0.14499999E-04
VIN= 0.12000000E 02
VOUT= 0.62563016E 01
VCLTR= 0.17387080E 02
VBASE= 0.36735230E 01
VBE= -0.25827786E 01
VCE= 0.13713557E 02
CURL= -0.42095962E-03
CCLTR= 0.13713704E-07
CEMTR= -0.25827799E-08
CBASE= -0.16296484E-07

TIME= 0.14999999E-04
VIN= 0.12000000E 02
VOUT= 0.61328876E 01
VCLTR= 0.17162015E 02
VBASE= 0.36313912E 01
VBE= -0.25014964E 01
VCE= 0.13530624E 02
CURL= -0.68500530E-03
CCLTR= 0.13530771E-07
CEMTR= -0.25014977E-08

CBASE= -0.16032268E-07

TIME= 0.15499999E-04
VIN= 0.12000000E 02
VOUT= 0.59842432E 01
VCLTR= 0.16864434E 02
VBASE= 0.36055655E 01
VBE= -0.23786776E 01
VCE= 0.13258868E 02
CURL= -0.93595471E-03
CCLTR= 0.13259015E-07
CEMTR= -0.23786789E-08
CBASE= -0.15637694E-07

TIME= 0.15999999E-04
VIN= 0.12000000E 02
VOUT= 0.58118424E 01
VCLTR= 0.16498423E 02
VBASE= 0.35704658E 01
VBE= -0.22413766E 01
VCE= 0.12927958E 02
CURL= -0.11702986E-02
CCLTR= 0.12928105E-07
CEMTR= -0.22413779E-08
CBASE= -0.15169483E-07

TIME= 0.16499998E-04
VIN= 0.12000000E 02
VOUT= 0.56180070E 01
VCLTR= 0.16069724E 02
VBASE= 0.35371267E 01
VBE= -0.20808803E 01
VCE= 0.12532598E 02
CURL= -0.13847511E-02
CCLTR= 0.12532745E-07
CEMTR= -0.20808817E-08
CBASE= -0.14613627E-07

TIME= 0.16999999E-04
VIN= 0.12000000E 02
VOUT= 0.54052161E 01
VCLTR= 0.15584758E 02
VBASE= 0.35057623E 01
VBE= -0.18994538E 01
VCE= 0.12078995E 02
CURL= -0.15763332E-02
CCLTR= 0.12079142E-07
CEMTR= -0.18994552E-08
CBASE= -0.13978598E-07

TIME= 0.17499998E-04
VIN= 0.12000000E 02
VOUT= 0.51762070E 01
VCLTR= 0.15050715E 02
VBASE= 0.34758134E 01
VBE= -0.17003935E 01
VCE= 0.11574902E 02
CURL= -0.17424086E-02
CCLTR= 0.11575048E-07
CEMTR= -0.17003948E-08
CBASE= -0.13275443E-07

TIME= 0.17999998E-04
VIN= 0.12000000E 02
VOUT= 0.49339520E 01
VCLTR= 0.14475477E 02

VBASE= 0.34471536E 01
VBE= -0.14867983E 01
VCE= 0.11028324E 02
CURL= -0.18807179E-02
CCLTR= 0.11028471E-07
CEMTR= -0.14867996E-08
CBASE= -0.12515271E-07

TIME= 0.18499999E-04
VIN= 0.12000000E 02
VOUT= 0.46816111E 01
VCLTR= 0.13867491E 02
VBASE= 0.34197441E 01
VBE= -0.12618670E 01
VCE= 0.10447747E 02
CURL= -0.19894103E-02
CCLTR= 0.10447894E-07
CEMTR= -0.12618683E-08
CBASE= -0.11709762E-07

TIME= 0.18999998E-04
VIN= 0.12000000E 02
VOUT= 0.44224859E 01
VCLTR= 0.13235643E 02
VBASE= 0.33936471E 01
VBE= -0.10288388E 01
VCE= 0.98419959E 01
CURL= -0.20670691E-02
CCLTR= 0.98421431E-08
CEMTR= -0.10288401E-08
CBASE= -0.10870983E-07

TIME= 0.19499999E-04
VIN= 0.12000000E 02
VOUT= 0.41599670E 01
VCLTR= 0.12589126E 02
VBASE= 0.33686582E 01
VBE= -0.79130878E 00
VCE= 0.92204677E 01
CURL= -0.21127300E-02
CCLTR= 0.92206149E-08
CEMTR= -0.79131009E-09
CBASE= -0.10011925E-07

TIME= 0.19999998E-04
VIN= 0.12000000E 02
VOUT= 0.38974919E 01
VCLTR= 0.11937311E 02
VBASE= 0.33446172E 01
VBE= -0.55287466E 00
VCE= 0.85926946E 01
CURL= -0.21258934E-02
CCLTR= 0.85928417E-08
CEMTR= -0.55287598E-09
CBASE= -0.91457177E-08

TIME= 0.20499998E-04
VIN= 0.12000000E 02
VOUT= 0.36384933E 01
VCLTR= 0.11289611E 02
VBASE= 0.33211531E 01
VBE= -0.31734020E-00
VCE= 0.79684588E 01
CURL= -0.21065295E-02
CCLTR= 0.79686060E-08
CEMTR= -0.31734153E-09

CBASE= -0.82859474E-08

TIME= 0.20999998E-04
VIN= 0.12000000E 02
VOUT= 0.33863514E 01
VCLTR= 0.10655341E 02
VBASE= 0.32974236E 01
VBE= -0.98927855E-01
VCE= 0.73579171E 01
CURL= -0.20550781E-02
CCLTR= 0.73580692E-08
CEMTR= -0.88924331E-10
CBASE= -0.74469935E-08

TIME= 0.21499998E-04
VIN= 0.12000000E 02
VOUT= 0.31443587E 01
VCLTR= 0.10043595E 02
VBASE= 0.32723870E 01
VBE= 0.12802827E-00
VCE= 0.67712082E 01
CURL= -0.19724395E-02
CCLTR= 0.67916235E-08
CEMTR= 0.14847901E-09
CBASE= -0.66431444E-08

TIME= 0.21999998E-04
VIN= 0.12000000E 02
VOUT= 0.29156715E 01
VCLTR= 0.94630731E 01
VBASE= 0.32437616E 01
VBE= 0.32809001E-00
VCE= 0.62193114E 01
CURL= -0.18599626E-02
CCLTR= 0.50740903E-07
CEMTR= 0.45253865E-07
CBASE= -0.54870375E-08

TIME= 0.22499998E-04
VIN= 0.12000000E 02
VOUT= 0.27033006E 01
VCLTR= 0.89195644E 01
VBASE= 0.32068034E 01
VBE= 0.50350279E 00
VCE= 0.57127608E 01
CURL= -0.17194472E-02
CCLTR= 0.37902299E-04
CEMTR= 0.38241257E-04
CBASE= 0.33895821E-06

TIME= 0.22999998E-04
VIN= 0.12000000E 02
VOUT= 0.25102934E 01
VCLTR= 0.82621781E 01
VBASE= 0.31219653E 01
VBE= 0.61167189E 00
VCE= 0.51402128E 01
CURL= -0.15512625E-02
CCLTR= 0.24289467E-02
CEMTR= 0.24510012E-02
CBASE= 0.22054504E-04

TIME= 0.23499998E-04
VIN= 0.12000000E 02
VOUT= 0.23412760E 01
VCLTR= 0.70489706E 01

VBASE= 0.29743247E 01
VBE= 0.63304879E 00
VCE= 0.40746457E 01
CURL= -0.13361819E-02
CCLTR= 0.55269974E-02
CEMTR= 0.55771887E-02
CBASE= 0.50191243E-04

TIME= 0.23999998E-04
VIN= 0.12000000E 02
VOUT= 0.22025760E 01
VCLTR= 0.55318344E 01
VBASE= 0.28413735E 01
VBE= 0.63879755E 00
VCE= 0.26904609E 01
CURL= -0.10513938E-02
CCLTR= 0.68946838E-02
CEMTR= 0.69572974E-02
CBASE= 0.62613631E-04

TIME= 0.24499997E-04
VIN= 0.12000000E 02
VOUT= 0.21014028E 01
VCLTR= 0.39425435E 01
VBASE= 0.27421588E 01
VBE= 0.64075603E 00
VCE= 0.12003846E 01
CURL= -0.68820463E-03
CCLTR= 0.74340916E-02
CEMTR= 0.75016056E-02
CBASE= 0.67513960E-04

TIME= 0.24999998E-04
VIN= 0.12000000E 02
VOUT= 0.20451105E 01
VCLTR= 0.24666576E 01
VBASE= 0.26835150E 01
VBE= 0.63840441E 00
VCE= -0.21685734E-00
CURL= -0.24746296E-03
CCLTR= 0.67912095E-02
CEMTR= 0.68528870E-02
CBASE= 0.61677478E-04

TIME= 0.25499997E-04
VIN= 0.12000000E 02
VOUT= 0.20404939E 01
VCLTR= 0.20679531E 01
VBASE= 0.26095951E 01
VBE= 0.56910126E 00
VCE= -0.54164213E 00
CURL= 0.24589244E-03
CCLTR= 0.14380322E-03
CEMTR= 0.31240537E-03
CBASE= 0.16860214E-03

TIME= 0.25999998E-04
VIN= 0.12000000E 02
VOUT= 0.20883548E 01
VCLTR= 0.21374872E 01
VBASE= 0.26750350E 01
VBE= 0.58668026E 00
VCE= -0.53754786E 00
CURL= 0.74095041E-03
CCLTR= 0.64815177E-03
CEMTR= 0.79695950E-03

CRASE= 0.14880773E-03

TIME= 0.26499998E-04
VIN= 0.12000000E 02
VOUT= 0.21865162E 01
VCLTR= 0.22496666E 01
VBASE= 0.27840269E 01
VBE= 0.59751077E 00
VCE= -0.53436026E 00
CURL= 0.12314621E-02
CCLTR= 0.11605337E-02
CEMTR= 0.12975043E-02
CBASE= 0.13697059E-03

TIME= 0.26999997E-04
VIN= 0.12000000E 02
VOUT= 0.23331860E 01
VCLTR= 0.24089594E 01
VBASE= 0.29376767E 01
VBE= 0.60449070E 00
VCE= -0.52871726E 00
CURL= 0.17151985E-02
CCLTR= 0.16428478E-02
CEMTR= 0.17595321E-02
CBASE= 0.11668430E-03

TIME= 0.27499998E-04
VIN= 0.12000000E 02
VOUT= 0.25263172E 01
VCLTR= 0.26164573E 01
VBASE= 0.31358297E 01
VBE= 0.60951255E 00
VCE= -0.51937250E 00
CURL= 0.21897683E-02
CCLTR= 0.20958194E-02
CEMTR= 0.21858936E-02
CBASE= 0.90074201E-04

TIME= 0.27999997E-04
VIN= 0.12000000E 02
VOUT= 0.27636459E 01
VCLTR= 0.28754059E 01
VBASE= 0.33766662E 01
VBE= 0.61302041E 00
VCE= -0.50126034E 00
CURL= 0.26527017E-02
CCLTR= 0.24887148E-02
CEMTR= 0.25467142E-02
CBASE= 0.57999510E-04

TIME= 0.28499997E-04
VIN= 0.12000000E 02
VOUT= 0.30426956E 01
VCLTR= 0.32068304E 01
VBASE= 0.36560849E 01
VBE= 0.61338937E 00
VCE= -0.44925455E-00
CURL= 0.31010707E-02
CCLTR= 0.25853992E-02
CEMTR= 0.26136695E-02
CBASE= 0.28270326E-04

TIME= 0.28999997E-04
VIN= 0.12000000E 02
VOUT= 0.33607047E 01
VCLTR= 0.37261540E 01

VBASE= 0.39623872E 01
VBE= 0.60168250E 00
VCE= -0.23623327E-00
CURL= 0.35289414E-02
CCLTR= 0.16540827E-02
CEMTR= 0.16691069E-02
CBASE= 0.15024125E-04

TIME= 0.29499997E-04
VIN= 0.12000000E 02
VOUT= 0.37141510E 01
VCLTR= 0.45818473E 01
VBASE= 0.42767008E 01
VBE= 0.56254985E 00
VCE= 0.30514651E-00
CURL= 0.39226835E-02
CCLTR= 0.36719847E-03
CEMTR= 0.37053353E-03
CBASE= 0.33350588E-05

TIME= 0.29999997E-04
VIN= 0.12000000E 02
VOUT= 0.40984328E 01
VCLTR= 0.56964456E 01
VBASE= 0.45403532E 01
VBE= 0.44192035E-00
VCE= 0.11560924E 01
CURL= 0.42663655E-02
CCLTR= 0.35488766E-05
CEMTR= 0.35803817E-05
CBASE= 0.31505152E-07

TIME= 0.30499997E-04
VIN= 0.12000000E 02
VOUT= 0.45087028E 01
VCLTR= 0.69143753E 01
VBASE= 0.46937252E 01
VBE= 0.18502237E-00
VCE= 0.22206501E 01
CURL= 0.45514161E-02
CCLTR= 0.24022736E-08
CEMTR= 0.36814533E-09
CBASE= -0.20341282E-08

TIME= 0.30999997E-04
VIN= 0.12000000E 02
VOUT= 0.49400661E 01
VCLTR= 0.81999709E 01
VBASE= 0.47302057E 01
VBE= -0.20986035E-00
VCE= 0.34697652E 01
CURL= 0.47740100E-02
CCLTR= 0.34699125E-08
CEMTR= -0.20986164E-09
CBASE= -0.36797741E-08

TIME= 0.31499997E-04
VIN= 0.12000000E 02
VOUT= 0.53871014E 01
VCLTR= 0.95363352E 01
VBASE= 0.46851145E 01
VBE= -0.70198690E 00
VCE= 0.48512206E 01
CURL= 0.49306907E-02
CCLTR= 0.48513678E-08
CEMTR= -0.70198823E-09

CBASE= -0.55533560E-08

TIME= 0.31999996E-04
VIN= 0.12000000E 02
VOUT= 0.58434208E 01
VCLTR= 0.10905376E 02
VBASE= 0.46051799E 01
VBE= -0.12382407E 01
VCE= 0.63002166E 01
CURL= 0.50197385E-02
CCLTR= 0.63003638E-08
CEMTR= -0.12382421E-08
CBASE= -0.75386059E-08

TIME= 0.32499996E-04
VIN= 0.12000000E 02
VOUT= 0.63019743E 01
VCLTR= 0.12287357E 02
VBASE= 0.45211308E 01
VBE= -0.17808434E 01
VCE= 0.77662261E 01
CURL= 0.50399324E-02
CCLTR= 0.77663733E-08
CEMTR= -0.17808448E-08
CBASE= -0.95472181E-08

TIME= 0.32999996E-04
VIN= 0.12000000E 02
VOUT= 0.67553902E 01
VCLTR= 0.13662118E 02
VBASE= 0.44452956E 01
VBE= -0.23100945E 01
VCE= 0.92168222E 01
CURL= 0.49911235E-02
CCLTR= 0.92169694E-08
CEMTR= -0.23100959E-08
CBASE= -0.11527065E-07

TIME= 0.33499996E-04
VIN= 0.12000000E 02
VOUT= 0.71963175E 01
VCLTR= 0.15009619E 02
VBASE= 0.43781814E 01
VBE= -0.28181360E 01
VCE= 0.10631438E 02
CURL= 0.48741754E-02
CCLTR= 0.10631584E-07
CEMTR= -0.28181373E-08
CBASE= -0.13449722E-07

TIME= 0.33999995E-04
VIN= 0.12000000E 02
VOUT= 0.76176543E 01
VCLTR= 0.16310297E 02
VBASE= 0.43171315E 01
VBE= -0.33005226E 01
VCE= 0.11993165E 02
CURL= 0.46909428E-02
CCLTR= 0.11993312E-07
CEMTR= -0.33005240E-08
CBASE= -0.15293837E-07

TIME= 0.34499995E-04
VIN= 0.12000000E 02
VOUT= 0.80126619E 01
VCLTR= 0.17545390E 02

VBASE= 0.42591867E 01
VBE= -0.37534751E 01
VCE= 0.13286204E 02
CURL= 0.44442397E-02
CCLTR= 0.13286351E-07
CEMTR= -0.37534764E-08
CBASE= -0.17039827E-07

TIME= 0.34999995E-04
VIN= 0.12000000E 02
VOUT= 0.83750607E 01
VCLTR= 0.18697216E 02
VBASE= 0.42027127E 01
VBE= -0.41723479E 01
VCE= 0.14494503E 02
CURL= 0.41377924E-02
CCLTR= 0.14494651E-07
CEMTR= -0.41723493E-08
CBASE= -0.18667000E-07

TIME= 0.35499994E-04
VIN= 0.12000000E 02
VOUT= 0.86991302E 01
VCLTR= 0.19749435E 02
VBASE= 0.41496263E 01
VBE= -0.45495038E 01
VCE= 0.15599808E 02
CURL= 0.37761827E-02
CCLTR= 0.15599956E-07
CEMTR= -0.45495051E-08
CBASE= -0.20149460E-07

TIME= 0.35999994E-04
VIN= 0.12000000E 02
VOUT= 0.89797083E 01
VCLTR= 0.20687174E 02
VBASE= 0.40990855E 01
VBE= -0.48806228E 01
VCE= 0.16588089E 02
CURL= 0.33647694E-02
CCLTR= 0.16588236E-07
CEMTR= -0.48806242E-08
CBASE= -0.21468860E-07

TIME= 0.36499994E-04
VIN= 0.12000000E 02
VOUT= 0.92121310E 01
VCLTR= 0.21497046E 02
VBASE= 0.40349323E 01
VBE= -0.51771986E 01
VCE= 0.17462114E 02
CURL= 0.29096070E-02
CCLTR= 0.17462261E-07
CEMTR= -0.51772000E-08
CBASE= -0.22639461E-07

TIME= 0.36999994E-04
VIN= 0.12000000E 02
VOUT= 0.93928216E 01
VCLTR= 0.22168081E 02
VBASE= 0.39741746E 01
VBE= -0.54186469E 01
VCE= 0.18193906E 02
CURL= 0.24173774E-02
CCLTR= 0.18194053E-07
CEMTR= -0.54186483E-08

CBASE= -0.2361270E-07

TIME= 0.37499993E-04
VIN= 0.12000000E 02
VOUT= 0.95186046E 01
VCLTR= 0.22690832E 02
VBASE= 0.38996030E 01
VBE= -0.56190015E 01
VCE= 0.18791229E 02
CURL= 0.18952549E-02
CCLTR= 0.18791375E-07
CEMTR= -0.56190028E-08
CBASE= -0.24410378E-07

TIME= 0.37999993E-04
VIN= 0.12000000E 02
VOUT= 0.95878407E 01
VCLTR= 0.23059063E 02
VBASE= 0.38627703E 01
VBE= -0.57250704E 01
VCE= 0.19196293E 02
CURL= 0.13508543E-02
CCLTR= 0.19196440E-07
CEMTR= -0.57250718E-08
CBASE= -0.24921511E-07

TIME= 0.38499992E-04
VIN= 0.12000000E 02
VOUT= 0.95986524E 01
VCLTR= 0.23267274E 02
VBASE= 0.38129332E 01
VBE= -0.57857192E 01
VCE= 0.19454341E 02
CURL= 0.79203091E-03
CCLTR= 0.19454487E-07
CEMTR= -0.57857205E-08
CBASE= -0.25240207E-07

TIME= 0.38999993E-04
VIN= 0.12000000E 02
VOUT= 0.95504019E 01
VCLTR= 0.23312838E 02
VBASE= 0.37351861E 01
VBE= -0.58152158E 01
VCE= 0.19577651E 02
CURL= 0.22683998E-03
CCLTR= 0.19577799E-07
CEMTR= -0.58152171E-08
CBASE= -0.25393016E-07

TIME= 0.39499992E-04
VIN= 0.12000000E 02
VOUT= 0.94443043E 01
VCLTR= 0.23196897E 02
VBASE= 0.37103318E 01
VBE= -0.57339725E 01
VCE= 0.19486565E 02
CURL= -0.33655633E-03
CCLTR= 0.19486713E-07
CEMTR= -0.57339738E-08
CBASE= -0.25220686E-07

TIME= 0.39999992E-04
VIN= 0.12000000E 02
VOUT= 0.92805702E 01
VCLTR= 0.22920302E 02

VBASE= 0.36408656E 01
VBE= -0.56397046E 01
VCE= 0.19279437E 02
CURL= -0.89014029E-03
CCLTR= 0.19279584E-07
CEMTR= -0.56397059E-08
CHASE= -0.24919290E-07

TIME= 0.40499991E-04
VIN= 0.12000000E 02
VOUT= 0.90617690E 01
VCLTR= 0.22488384E 02
VBASE= 0.35764673E 01
VBE= -0.54853018E 01
VCE= 0.18911917E 02
CURL= -0.14259931E-02
CCLTR= 0.18912064E-07
CEMTR= -0.54853031E-08
CBASE= -0.24397366E-07

TIME= 0.40999991E-04
VIN= 0.12000000E 02
VOUT= 0.87907948E 01
VCLTR= 0.21908028E 02
VBASE= 0.35211908E 01
VBE= -0.52696040E 01
VCE= 0.18386837E 02
CURL= -0.19365055E-02
CCLTR= 0.18386985E-07
CEMTR= -0.52696053E-08
CBASE= -0.23656590E-07

TIME= 0.41499991E-04
VIN= 0.12000000E 02
VOUT= 0.84711722E 01
VCLTR= 0.21188072E 02
VBASE= 0.34676772E 01
VBE= -0.50034951E 01
VCE= 0.17720395E 02
CURL= -0.24144687E-02
CCLTR= 0.17720542E-07
CEMTR= -0.50034964E-08
CBASE= -0.22724038E-07

TIME= 0.41999990E-04
VIN= 0.12000000E 02
VOUT= 0.81071958E 01
VCLTR= 0.20339439E 02
VBASE= 0.34154870E 01
VBE= -0.46917088E 01
VCE= 0.16923952E 02
CURL= -0.28531674E-02
CCLTR= 0.16924099E-07
CEMTR= -0.46917102E-08
CBASE= -0.21615809E-07

TIME= 0.42499990E-04
VIN= 0.12000000E 02
VOUT= 0.77037977E 01
VCLTR= 0.19374878E 02
VBASE= 0.33646633E 01
VBE= -0.43391344E 01
VCE= 0.16010215E 02
CURL= -0.32464795E-02
CCLTR= 0.16010362E-07
CEMTR= -0.43391357E-08

CBASE= -0.20349497E-07

TIME= 0.42999990E-04
VIN= 0.12000000E 02
VOUT= 0.72664697E 01
VCLTR= 0.18308766E 02
VBASE= 0.33153287E 01
VBE= -0.39511409E 01
VCE= 0.14993437E 02
CURL= -0.35889614E-02
CCLTR= 0.14993584E-07
CEMTR= -0.39511422E-08
CBASE= -0.18944726E-07

TIME= 0.43499990E-04
VIN= 0.12000000E 02
VOUT= 0.68011790E 01
VCLTR= 0.17156889E 02
VBASE= 0.32674140E 01
VBE= -0.35337650E 01
VCE= 0.13889475E 02
CURL= -0.38759255E-02
CCLTR= 0.13889623E-07
CEMTR= -0.35337663E-08
CBASE= -0.17423389E-07

TIME= 0.43999989E-04
VIN= 0.12000000E 02
VOUT= 0.63142912E 01
VCLTR= 0.15936225E 02
VBASE= 0.32214842E 01
VBE= -0.30928070E 01
VCE= 0.12714741E 02
CURL= -0.41035033E-02
CCLTR= 0.12714887E-07
CEMTR= -0.30928083E-08
CBASE= -0.15807696E-07

TIME= 0.44499990E-04
VIN= 0.12000000E 02
VOUT= 0.58124509E 01
VCLTR= 0.14664654E 02
VBASE= 0.31765316E 01
VBE= -0.26359192E 01
VCE= 0.11488122E 02
CURL= -0.42686991E-02
CCLTR= 0.11488269E-07
CEMTR= -0.26359206E-08
CBASE= -0.14124189E-07

TIME= 0.44999989E-04
VIN= 0.12000000E 02
VOUT= 0.53025284E 01
VCLTR= 0.13360759E 02
VBASE= 0.31332538E 01
VBE= -0.21892745E 01
VCE= 0.10227504E 02
CURL= -0.43694296E-02
CCLTR= 0.10227652E-07
CEMTR= -0.21692758E-08
CBASE= -0.12396928E-07

TIME= 0.45499989E-04
VIN= 0.12000000E 02
VOUT= 0.47914836E 01
VCLTR= 0.12043501E 02

VBASE= 0.30909934E 01
VBE= -0.17004902E 01
VCE= 0.89525079E 01
CURL= -0.44045507E-02
CCLTR= 0.89526551E-08
CEMTR= -0.17004915E-08
CBASE= -0.10653146E-07

TIME= 0.45999988E-04
VIN= 0.12000000E 02
VOUT= 0.42862593E 01
VCLTR= 0.10731936E 02
VBASE= 0.30473500E 01
VBE= -0.12389092E 01
VCE= 0.76845856E 01
CURL= -0.43738731E-02
CCLTR= 0.76847327E-08
CEMTR= -0.12389106E-08
CBASE= -0.89236432E-08

TIME= 0.46499988E-04
VIN= 0.12000000E 02
VOUT= 0.37937634E 01
VCLTR= 0.94450390E 01
VBASE= 0.30025650E 01
VBE= -0.79119834E 00
VCE= 0.64424739E 01
CURL= -0.42781564E-02
CCLTR= 0.64426212E-08
CEMTR= -0.79119966E-09
CBASE= -0.72338209E-08

TIME= 0.46999988E-04
VIN= 0.12000000E 02
VOUT= 0.33207068E 01
VCLTR= 0.82013439E 01
VBASE= 0.29543878E 01
VBE= -0.36631895E-00
VCE= 0.52469561E 01
CURL= -0.41190978E-02
CCLTR= 0.52471033E-08
CEMTR= -0.36632028E-09
CBASE= -0.56134236E-08

TIME= 0.47499987E-04
VIN= 0.12000000E 02
VOUT= 0.28735323E 01
VCLTR= 0.70186859E 01
VBASE= 0.28972090E 01
VBE= 0.23676741E-01
VCE= 0.41214769E 01
CURL= -0.38993082E-02
CCLTR= 0.41219904E-08
CEMTR= 0.24044965E-10
CBASE= -0.40979455E-08

TIME= 0.47999987E-04
VIN= 0.12000000E 02
VOUT= 0.24584159E 01
VCLTR= 0.59140222E 01
VBASE= 0.28223323E 01
VBE= 0.36391654E-00
VCE= 0.30916899E 01
CURL= -0.36222671E-02
CCLTR= 0.17969657E-06
CEMTR= 0.17857254E-06

CBASE= -0.11240430E-08

TIME= 0.48499987E-04
VIN= 0.12000000E 02
VOUT= 0.20813318E 01
VCLTR= 0.48570393E 01
VBASE= 0.26901232E 01
VBE= 0.60879146E 00
VCE= 0.21669161E 01
CURL= -0.32919226E-02
CCLTR= 0.21742219E-02
CEMTR= 0.21939661E-02
CBASE= 0.19744126E-04

TIME= 0.48999987E-04
VIN= 0.12000000E 02
VOUT= 0.17500041E 01
VCLTR= 0.29316566E 01
VBASE= 0.23917467E 01
VBE= 0.64174262E 00
VCE= 0.53990984E 00
CURL= -0.28903949E-02
CCLTR= 0.77216035E-02
CEMTR= 0.77917293E-02
CBASE= 0.70125737E-04

TIME= 0.49499986E-04
VIN= 0.12000000E 02
VOUT= 0.14753257E 01
VCLTR= 0.13820509E 01
VBASE= 0.19457681E 01
VBE= 0.47044239E-00
VCE= -0.56371719E 00
CURL= -0.23882625E-02
CCLTR= -0.75749506E-03
CEMTR= -0.37333725E-03
CBASE= 0.38415781E-03

TIME= 0.49999986E-04
VIN= 0.12000000E 02
VOUT= 0.12671825E 01
VCLTR= 0.92043361E 00
VBASE= 0.14839204E 01
VBE= 0.21673798E-00
VCE= -0.56348684E 00
CURL= -0.18451126E-02
CCLTR= -0.76134510E-03
CEMTR= -0.38067173E-03
CBASE= 0.38067336E-03

TIME= 0.50499986E-04
VIN= 0.12000000E 02
VOUT= 0.11255458E 01
VCLTR= 0.63276174E 00
VBASE= 0.11967021E 01
VBE= 0.71156327E-01
VCE= -0.56394044E 00
CURL= -0.12832140E-02
CCLTR= -0.77474489E-03
CEMTR= -0.38737209E-03
CBASE= 0.38737279E-03

TIME= 0.50999986E-04
VIN= 0.12000000E 02
VOUT= 0.10485762E 01
VCLTR= 0.51599313E 00

VHASE= 0.10799628E 01
VBE= 0.31386627E-01
VCE= -0.56396973E 00
CURL= -0.71123395E-03
CCLTR= -0.77561816E-03
CEMTR= -0.38780876E-03
CBASE= 0.38780939E-03

TIME= 0.51499985E-04
VIN= 0.12000000E 02
VOUT= 0.10336582E 01
VCLTR= 0.56152547E 00
VBASE= 0.11236610E 01
VBE= 0.90002757E-01
VCE= -0.56213555E 00
CURL= -0.13751960E-03
CCLTR= -0.72278727E-03
CEMTR= -0.36139326E-03
CBASE= 0.36139400E-03

TIME= 0.51999985E-04
VIN= 0.12000000E 02
VOUT= 0.10775178E 01
VCLTR= 0.75747078E 00
VHASE= 0.13138828E 01
VBE= 0.23636501E-00
VCE= -0.55641209E 00
CURL= 0.43011423E-03
CCLTR= -0.57997185E-03
CEMTR= -0.28998475E-03
CBASE= 0.28998710E-03

TIME= 0.52499985E-04
VIN= 0.12000000E 02
VOUT= 0.11763631E 01
VCLTR= 0.10864930E 01
VBASE= 0.16371051E 01
VBE= 0.46074205E-00
VCE= -0.55061214E 00
CURL= 0.98453783E-03
CCLTR= -0.45669347E-03
CEMTR= -0.22462098E-03
CBASE= 0.23207248E-03

TIME= 0.52999984E-04
VIN= 0.12000000E 02
VOUT= 0.13266193E 01
VCLTR= 0.13819403E 01
VBASE= 0.19333991E 01
VBE= 0.60677983E 00
VCE= -0.55145887E 00
CURL= 0.15214834E-02
CCLTR= 0.15329720E-02
CEMTR= 0.17909337E-02
CBASE= 0.25796167E-03

TIME= 0.53499984E-04
VIN= 0.12000000E 02
VOUT= 0.15280367E 01
VCLTR= 0.15913360E 01
VBASE= 0.21395903E 01
VBE= 0.61155362E 00
VCE= -0.54825430E 00
CURL= 0.20473879E-02
CCLTR= 0.19941350E-02
CEMTR= 0.22279862E-02

CBASE= 0.23385122E-03

TIME= 0.53999984E-04
VIN= 0.12000000E 02
VOUT= 0.17788133E 01
VCLTR= 0.18517219E 01
VBASE= 0.23946089E 01
VBE= 0.61579563E 00
VCE= -0.54288706E 00
CURL= 0.25615154E-02
CCLTR= 0.25016771E-02
CEMTR= 0.26998983E-02
CBASE= 0.19822112E-03

TIME= 0.54499984E-04
VIN= 0.12000000E 02
VOUT= 0.20763250E 01
VCLTR= 0.21615957E 01
VBASE= 0.26958714E 01
VBE= 0.61954645E 00
VCE= -0.53427580E 00
CURL= 0.30614068E-02
CCLTR= 0.30406001E-02
CEMTR= 0.31942350E-02
CBASE= 0.15363484E-03

TIME= 0.54999983E-04
VIN= 0.12000000E 02
VOUT= 0.24177548E 01
VCLTR= 0.25181998E 01
VBASE= 0.30394968E 01
VBE= 0.62174211E 00
VCE= -0.52129709E 00
CURL= 0.35446410E-02
CCLTR= 0.34276179E-02
CEMTR= 0.35352453E-02
CBASE= 0.10762736E-03

TIME= 0.55499983E-04
VIN= 0.12000000E 02
VOUT= 0.28001057E 01
VCLTR= 0.29325285E 01
VBASE= 0.34231623E 01
VBE= 0.62305662E 00
VCE= -0.49063375E-00
CURL= 0.40086608E-02
CCLTR= 0.37172233E-02
CEMTR= 0.37745043E-02
CBASE= 0.57281082E-04

TIME= 0.55999982E-04
VIN= 0.12000000E 02
VOUT= 0.32201686E 01
VCLTR= 0.34837630E 01
VBASE= 0.38372432E 01
VBE= 0.61707472E 00
VCE= -0.35348019E-00
CURL= 0.44492764E-02
CCLTR= 0.29897133E-02
CEMTR= 0.30169865E-02
CBASE= 0.27273170E-04

TIME= 0.56499983E-04
VIN= 0.12000000E 02
VOUT= 0.36741734E 01
VCLTR= 0.44198699E 01

VBASE= 0.42625234E 01
VBE= 0.58835001E 00
VCE= 0.15734653E-00
CURL= 0.48537014E-02
CCLTR= 0.99050578E-03
CEMTR= 0.99950174E-03
CBASE= 0.89959648E-05

TIME= 0.56999982E-04
VIN= 0.12000000E 02
VOUT= 0.41569827E 01
VCLTR= 0.57619724E 01
VBASE= 0.46381389E 01
VBE= 0.48115624E-00
VCE= 0.11238334E 01
CURL= 0.52002684E-02
CCLTR= 0.16045818E-04
CEMTR= 0.16190889E-04
CBASE= 0.14507099E-06

TIME= 0.57499982E-04
VIN= 0.12000000E 02
VOUT= 0.46629334E 01
VCLTR= 0.72457750E 01
VBASE= 0.48605818E 01
VBE= 0.19764840E-00
VCE= 0.23851931E 01
CURL= 0.54753771E-02
CCLTR= 0.26802713E-08
CEMTR= 0.49525645E-09
CBASE= -0.21850149E-08

TIME= 0.57999981E-04
VIN= 0.12000000E 02
VOUT= 0.51866115E 01
VCLTR= 0.87912825E 01
VBASE= 0.49071616E 01
VBE= -0.27944983E-00
VCE= 0.38841209E 01
CURL= 0.56746633E-02
CCLTR= 0.38842682E-08
CEMTR= -0.27945116E-09
CBASE= -0.41637193E-08

TIME= 0.58499981E-04
VIN= 0.12000000E 02
VOUT= 0.57220132E 01
VCLTR= 0.10378355E 02
VBASE= 0.48333576E 01
VBE= -0.88865560E 00
VCE= 0.55449972E 01
CURL= 0.57955515E-02
CCLTR= 0.55451445E-08
CEMTR= -0.88865693E-09
CBASE= -0.64338014E-08

TIME= 0.58999981E-04
VIN= 0.12000000E 02
VOUT= 0.62618704E 01
VCLTR= 0.11985470E 02
VBASE= 0.47159545E 01
VBE= -0.15459158E 01
VCE= 0.72695163E 01
CURL= 0.58364926E-02
CCLTR= 0.72696636E-08
CEMTR= -0.15459172E-08

CHASE= -0.88155808E-01

TIME= 0.59499981E-04
VIN= 0.12000000E 02
VOUT= 0.67979009E 01
VCLTR= 0.13589692E 02
VBASE= 0.46015096E 01
VBE= -0.21963912E 01
VCE= 0.89881827E 01
CURL= 0.57970522E-02
CCLTR= 0.89883299E-08
CEMTR= -0.21963926E-08
CBASE= -0.11184722E-07

TIME= 0.59999980E-04
VIN= 0.12000000E 02
VOUT= 0.73214867E 01
VCLTR= 0.15167588E 02
VBASE= 0.45040454E 01
VBE= -0.28174413E 01
VCE= 0.10663542E 02
CURL= 0.56779619E-02
CCLTR= 0.10663690E-07
CEMTR= -0.28174426E-08
CBASE= -0.13481132E-07

TIME= 0.60499980E-04
VIN= 0.12000000E 02
VOUT= 0.78241690E 01
VCLTR= 0.16696072E 02
VBASE= 0.44216080E 01
VBE= -0.34025609E 01
VCE= 0.12274464E 02
CURL= 0.54811174E-02
CCLTR= 0.12274612E-07
CEMTR= -0.34025622E-08
CBASE= -0.15677173E-07

TIME= 0.60999980E-04
VIN= 0.12000000E 02
VOUT= 0.82979033E 01
VCLTR= 0.18152936E 02
VBASE= 0.43484407E 01
VBE= -0.39494625E 01
VCE= 0.13804495E 02
CURL= 0.52095492E-02
CCLTR= 0.13804643E-07
CEMTR= -0.39494639E-08
CBASE= -0.17754106E-07

TIME= 0.61499979E-04
VIN= 0.12000000E 02
VOUT= 0.87351719E 01
VCLTR= 0.19517170E 02
VBASE= 0.42785896E 01
VBE= -0.44565821E 01
VCE= 0.15238580E 02
CURL= 0.48673730E-02
CCLTR= 0.15238728E-07
CEMTR= -0.44565835E-08
CBASE= -0.19695311E-07

TIME= 0.61999979E-04
VIN= 0.12000000E 02
VOUT= 0.91291902E 01

VCLTR= 0.20769422E 02
VBASE= 0.42176654E 01
VBE= -0.49115248E 01
VCE= 0.16551756E 02
CURL= 0.44597080E-02
CCLTR= 0.16551903E-07
CEMTR= -0.49115261E-08
CBASE= -0.21463429E-07

TIME= 0.62499978E-04
VIN= 0.12000000E 02
VOUT= 0.94736460E 01
VCLTR= 0.21891760E 02
VBASE= 0.41527963E 01
VBE= -0.53208496E 01
VCE= 0.17738964E 02
CURL= 0.39926067E-02
CCLTR= 0.17739110E-07
CEMTR= -0.53208509E-08
CBASE= -0.23059961E-07

TIME= 0.62999977E-04
VIN= 0.12000000E 02
VOUT= 0.97633472E 01
VCLTR= 0.22868723E 02
VBASE= 0.41045830E 01
VBE= -0.56587642E 01
VCE= 0.18764140E 02
CURL= 0.34729761E-02
CCLTR= 0.18764287E-07
CEMTR= -0.56587655E-08
CBASE= -0.24423052E-07

TIME= 0.63499977E-04
VIN= 0.12000000E 02
VOUT= 0.99933197E 01
VCLTR= 0.23686156E 02
VBASE= 0.40353855E 01
VBE= -0.59579341E 01
VCE= 0.19650770E 02
CURL= 0.29084262E-02
CCLTR= 0.19650918E-07
CEMTR= -0.59579354E-08
CBASE= -0.25608853E-07

TIME= 0.63999975E-04
VIN= 0.12000000E 02
VOUT= 0.10159959E 02
VCLTR= 0.24332941E 02
VBASE= 0.39522126E 01
VBE= -0.62077462E 01
VCE= 0.20380728E 02
CURL= 0.23072156E-02
CCLTR= 0.20380875E-07
CEMTR= -0.62077475E-08
CBASE= -0.26588622E-07

TIME= 0.64499975E-04
VIN= 0.12000000E 02
VOUT= 0.10260959E 02
VCLTR= 0.24800990E 02
VBASE= 0.38988050E 01

VBE= -0.63621540E 01
VCE= 0.20902184E 02
CURL= 0.16781145E-02
CCLTR= 0.20902331E-07
CEMTR= -0.63621553E-08
CBASE= -0.27264487E-07

TIME= 0.64999975E-04
VIN= 0.12000000E 02
VOUT= 0.10293852E 02
VCLTR= 0.25083216E 02
VBASE= 0.38067830E 01
VBE= -0.64870693E 01
VCE= 0.21276433E 02
CURL= 0.10302134E-02
CCLTR= 0.21276580E-07
CEMTR= -0.64870706E-08
CBASE= -0.27763651E-07

TIME= 0.65499973E-04
VIN= 0.12000000E 02
VOUT= 0.10258845E 02
VCLTR= 0.25177639E 02
VBASE= 0.37747401E 01
VBE= -0.64841051E 01
VCE= 0.21402898E 02
CURL= 0.37291683E-03
CCLTR= 0.21403045E-07
CEMTR= -0.64841064E-08
CBASE= -0.27887152E-07

TIME= 0.65999973E-04
VIN= 0.12000000E 02
VOUT= 0.10155127E 02
VCLTR= 0.25082131E 02
VBASE= 0.36991986E 01
VBE= -0.64559290E 01
VCE= 0.21382932E 02
CURL= -0.28436225E-03
CCLTR= 0.21383079E-07
CEMTR= -0.64559303E-08
CBASE= -0.27839009E-07

TIME= 0.66499972E-04
VIN= 0.12000000E 02
VOUT= 0.99844475E 01
VCLTR= 0.24799541E 02
VBASE= 0.36334453E 01
VBE= -0.63510022E 01
VCE= 0.21166096E 02
CURL= -0.93217495E-03
CCLTR= 0.21166243E-07
CEMTR= -0.63510035E-08
CBASE= -0.27517246E-07

TIME= 0.66999971E-04
VIN= 0.12000000E 02
VOUT= 0.97489294E 01
VCLTR= 0.24334567E 02
VBASE= 0.35707596E 01
VBE= -0.61781698E 01
VCE= 0.20763807E 02
CURL= -0.15612731E-02
CCLTR= 0.20763955E-07
CEMTR= -0.61781711E-08
CBASE= -0.26942125E-07

TIME= 0.67499971E-04
VIN= 0.12000000E 02
VOUT= 0.94516794E 01
VCLTR= 0.23694585E 02
VBASE= 0.35089000E 01
VBE= -0.59427794E 01
VCE= 0.20185684E 02
CURL= -0.21627110E-02
CCLTR= 0.20185831E-07
CEMTR= -0.59427807E-08
CBASE= -0.26128612E-07

TIME= 0.67999970E-04
VIN= 0.12000000E 02
VOUT= 0.90967176E 01
VCLTR= 0.22889511E 02
VBASE= 0.34481194E 01
VBE= -0.56485982E 01
VCE= 0.19441392E 02
CURL= -0.27279766E-02
CCLTR= 0.19441538E-07
CEMTR= -0.56485996E-08
CBASE= -0.25090139E-07

TIME= 0.68499969E-04
VIN= 0.12000000E 02
VOUT= 0.86888916E 01
VCLTR= 0.21931612E 02
VBASE= 0.33886344E 01
VBE= -0.53002571E 01
VCE= 0.18542977E 02
CURL= -0.32491119E-02
CCLTR= 0.18543125E-07
CEMTR= -0.53002585E-08
CBASE= -0.23843383E-07

TIME= 0.68999968E-04
VIN= 0.12000000E 02
VOUT= 0.92338031E 01
VCLTR= 0.20835317E 02
VBASE= 0.33305900E 01
VBE= -0.49032131E 01
VCE= 0.17504727E 02
CURL= -0.37188281E-02
CCLTR= 0.17504874E-07
CEMTR= -0.49032144E-08
CBASE= -0.22408088E-07

TIME= 0.69499968E-04
VIN= 0.12000000E 02
VOUT= 0.77377195E 01
VCLTR= 0.19616994E 02
VBASE= 0.32736283E 01
VBE= -0.44640912E 01
VCE= 0.16343366E 02
CURL= -0.41306072E-02
CCLTR= 0.16343513E-07
CEMTR= -0.44640925E-08
CBASE= -0.20807605E-07

TIME= 0.69999967E-04
VIN= 0.12000000E 02
VOUT= 0.72075031E 01
VCLTR= 0.18294737E 02
VBASE= 0.32188663E 01

VBE= -0.39886367E 01
VCE= 0.15075870E 02
CURL= -0.44787935E-02
CCLTR= 0.15076017E-07
CEMTR= -0.39886381E-08
CBASE= -0.19064655E-07

TIME= 0.70499966E-04
VIN= 0.12000000E 02
VOUT= 0.66504680E 01
VCLTR= 0.16888037E 02
VBASE= 0.31649891E 01
VBE= -0.34854789E 01
VCE= 0.13723049E 02
CURL= -0.47586723E-02
CCLTR= 0.13723195E-07
CEMTR= -0.34854802E-08
CBASE= -0.17208675E-07

TIME= 0.70999966E-04
VIN= 0.12000000E 02
VOUT= 0.60743222E 01
VCLTR= 0.15417570E 02
VBASE= 0.31124565E 01
VBE= -0.29618657E 01
VCE= 0.12305114E 02
CURL= -0.49665341E-02
CCLTR= 0.12305260E-07
CEMTR= -0.29618670E-08
CBASE= -0.15267128E-07

TIME= 0.71499965E-04
VIN= 0.12000000E 02
VOUT= 0.54870508E 01
VCLTR= 0.13904876E 02
VBASE= 0.30624650E 01
VBE= -0.24245858E 01
VCE= 0.10842410E 02
CURL= -0.50997223E-02
CCLTR= 0.10842558E-07
CEMTR= -0.24245872E-08
CBASE= -0.13267145E-07

TIME= 0.71999964E-04
VIN= 0.12000000E 02
VOUT= 0.48966938E 01
VCLTR= 0.12371906E 02
VBASE= 0.30093135E 01
VBE= -0.18873802E 01
VCE= 0.93625924E 01
CURL= -0.51566785E-02
CCLTR= 0.93627395E-08
CEMTR= -0.18873815E-08
CBASE= -0.11250121E-07

TIME= 0.72499963E-04
VIN= 0.12000000E 02
VOUT= 0.43114163E 01
VCLTR= 0.10840951E 02
VBASE= 0.29531934E 01
VBE= -0.13582234E 01
VCE= 0.78877578E 01
CURL= -0.51369472E-02
CCLTR= 0.78879049E-08
CEMTR= -0.13582247E-08
CBASE= -0.92461297E-08

TIME= 0.72999962E-04
VIN= 0.12000000E 02
VOUT= 0.37393592E 01
VCLTR= 0.93342671E 01
VBASE= 0.28956571E 01
VBE= -0.84370210E 00
VCE= 0.64386099E 01
CURL= -0.50411814E-02
CCLTR= 0.64387572E-08
CEMTR= -0.84370343E-09
CBASE= -0.72824606E-08

TIME= 0.73499962E-04
VIN= 0.12000000E 02
VOUT= 0.31883959E 01
VCLTR= 0.78735748E 01
VBASE= 0.28300703E 01
VBE= -0.35832558E-00
VCE= 0.50435044E 01
CURL= -0.48711404E-02
CCLTR= 0.50436516E-08
CEMTR= -0.35832691E-09
CBASE= -0.54019785E-08

TIME= 0.73999961E-04
VIN= 0.12000000E 02
VOUT= 0.26661914E 01
VCLTR= 0.64799444E 01
VBASE= 0.27480453E 01
VBE= 0.81853880E-01
VCE= 0.37318990E 01
CURL= -0.46296569E-02
CCLTR= 0.37354781E-08
CEMTR= 0.85315581E-10
CBASE= -0.36501626E-08

TIME= 0.74499960E-04
VIN= 0.12000000E 02
VOUT= 0.21801695E 01
VCLTR= 0.51734281E 01
VBASE= 0.26333147E 01
VBE= 0.45314512E-00
VCE= 0.25401134E 01
CURL= -0.43205902E-02
CCLTR= 0.54657005E-05
CEMTR= 0.55132283E-05
CBASE= 0.47527862E-07

TIME= 0.74999960E-04
VIN= 0.12000000E 02
VOUT= 0.17380137E 01
VCLTR= 0.36411545E 01
VBASE= 0.23720235E 01
VBE= 0.63400987E 00
VCE= 0.12691309E 01
CURL= -0.39452061E-02
CCLTR= 0.57351208E-02
CEMTR= 0.57872052E-02
CBASE= 0.52084273E-04

TIME= 0.75499959E-04
VIN= 0.12000000E 02
VOUT= 0.13507188E 01
VCLTR= 0.13434266E 01
VBASE= 0.19148389E 01

VBE= 0.56412013E-00
VCE= -0.57141238E-00
CURL= -0.34667631E-02
CCLTR= -0.64262488E-03
CEMTR= -0.12273988E-03
CBASE= 0.51728500E-03

TIME= 0.75999958E-04
VIN= 0.12000000E-02
VOUT= 0.10328157E-01
VCLTR= 0.60121991E-00
VBASE= 0.11697106E-01
VBE= 0.13689497E-00
VCE= -0.56849076E-00
CURL= -0.29146075E-02
CCLTR= -0.92292298E-03
CEMTR= -0.46146105E-03
CBASE= 0.46146192E-03

TIME= 0.76499958E-04
VIN= 0.12000000E-02
VOUT= 0.78846157E-00
VCLTR= 0.49613856E-01
VBASE= 0.62054204E-00
VBE= -0.16791952E-00
VCE= -0.57092819E-00
CURL= -0.23300635E-02
CCLTR= -0.10136299E-02
CEMTR= -0.50681484E-03
CBASE= 0.50581507E-03

TIME= 0.76999956E-04
VIN= 0.12000000E-02
VOUT= 0.61678517E-00
VCLTR= -0.30581076E-00
VBASE= 0.26647357E-00
VBE= -0.35031160E-00
VCE= -0.57228433E-00
CURL= -0.17228543E-02
CCLTR= -0.10679035E-02
CEMTR= -0.53395181E-03
CBASE= 0.53395168E-03

TIME= 0.77499956E-04
VIN= 0.12000000E-02
VOUT= 0.51593411E-00
VCLTR= -0.47098769E-00
VBASE= 0.10125253E-00
VBE= -0.41468158E-00
VCE= -0.57224022E-00
CURL= -0.11026626E-02
CCLTR= -0.10660930E-02
CEMTR= -0.53304662E-03
CBASE= 0.53304637E-03

TIME= 0.77999955E-04
VIN= 0.12000000E-02
VOUT= 0.48319840E-00
VCLTR= -0.45598403E-00
VBASE= 0.11612104E-00
VBE= -0.36707737E-00
VCE= -0.57210506E-00
CURL= -0.47876819E-03
CCLTR= -0.10605656E-02
CEMTR= -0.53028286E-03
CBASE= 0.53028270E-03

TIME= 0.78499954E-04
VIN= 0.12000000E 02
VOUT= 0.51511145E 00
VCLTR= -0.27255127E-00
VBASE= 0.29641835E-00
VBE= -0.21869309E-00
VCE= -0.56896961E 00
CURL= 0.14010584E-03
CCLTR= -0.94007853E-03
CEMTR= -0.47003920E-03
CBASE= 0.47003932E-03

TIME= 0.78999954E-04
VIN= 0.12000000E 02
VOUT= 0.60758591E 00
VCLTR= 0.63031010E-01
VBASE= 0.62777064E 00
VBE= 0.20184737E-01
VCE= -0.56473963E 00
CURL= 0.74594572E-03
CCLTR= -0.79892885E-03
CEMTR= -0.39946412E-03
CBASE= 0.39946473E-03

TIME= 0.79499953E-04
VIN= 0.12000000E 02
VOUT= 0.75603592E 00
VCLTR= 0.53316696E 00
VBASE= 0.10894384E 01
VBE= 0.33340257E-00
VCE= -0.55627153E 00
CURL= 0.13315620E-02
CCLTR= -0.57679172E-03
CEMTR= -0.28836744E-03
CBASE= 0.28842427E-03

TIME= 0.79999952E-04
VIN= 0.12000000E 02
VOUT= 0.95581353E 00
VCLTR= 0.10159381E 01
VBASE= 0.15681119E 01
VBE= 0.61229837E 00
VCE= -0.55217374E 00
CURL= 0.18915307E-02
CCLTR= 0.19954447E-02
CEMTR= 0.22644094E-02
CBASE= 0.26896471E-03

TIME= 0.80499952E-04
VIN= 0.12000000E 02
VOUT= 0.12064966E 01
VCLTR= 0.12697031E 01
VBASE= 0.18227282E 01
VBE= 0.61623164E 00
VCE= -0.55302513E 00
CURL= 0.24347027E-02
CCLTR= 0.23854170E-02
CEMTR= 0.26662733E-02
CBASE= 0.28085630E-03

TIME= 0.80999950E-04
VIN= 0.12000000E 02
VOUT= 0.15069710E 01
VCLTR= 0.15790167E 01
VBASE= 0.21267792E 01

VBE= 0.61980828E-00
VCE= -0.54776251E-00
CURL= 0.29636888E-02
CCLTR= 0.29055814E-02
CEMTR= 0.31436678E-02
CBASE= 0.23808644E-03

TIME= 0.81499950E-04
VIN= 0.12000000E-02
VOUT= 0.18543386E-01
VCLTR= 0.19370101E-01
VBASE= 0.24770594E-01
VBE= 0.62272084E-00
VCE= -0.54004936E-00
CURL= 0.34759869E-02
CCLTR= 0.34060405E-02
CEMTR= 0.35943285E-02
CBASE= 0.18828801E-03

TIME= 0.81999949E-04
VIN= 0.12000000E-02
VOUT= 0.22455196E-01
VCLTR= 0.23428615E-01
VBASE= 0.28705141E-01
VBE= 0.62499451E-00
VCE= -0.52765261E-00
CURL= 0.39691912E-02
CCLTR= 0.38627727E-02
CEMTR= 0.39955349E-02
CBASE= 0.13276219E-03

TIME= 0.82499948E-04
VIN= 0.12000000E-02
VOUT= 0.26772626E-01
VCLTR= 0.28021558E-01
VBASE= 0.33036230E-01
VBE= 0.62636040E-00
VCE= -0.50146723E-00
CURL= 0.44408261E-02
CCLTR= 0.42032654E-02
CEMTR= 0.42771188E-02
CBASE= 0.73853415E-04

TIME= 0.82999948E-04
VIN= 0.12000000E-02
VOUT= 0.31461257E-01
VCLTR= 0.33773280E-01
VBASE= 0.37682741E-01
VBE= 0.62214835E-00
VCE= -0.39094613E-00
CURL= 0.48872264E-02
CCLTR= 0.36332273E-02
CEMTR= 0.36667329E-02
CBASE= 0.33505581E-04

TIME= 0.83499947E-04
VIN= 0.12000000E-02
VOUT= 0.36481902E-01
VCLTR= 0.43409897E-01
VBASE= 0.42451769E-01
VBE= 0.59698675E-00
VCE= 0.95812850E-01
CURL= 0.52964605E-02
CCLTR= 0.13807716E-02
CEMTR= 0.13933119E-02
CBASE= 0.12540317E-04

TIME= 0.83999946E-04
VIN= 0.12000000E 02
VOUT= 0.41781102E 01
VCLTR= 0.57808400E 01
VBASE= 0.46751989E 01
VBE= 0.49708878E-00
VCE= 0.11056410E 01
CURL= 0.56448480E-02
CCLTR= 0.29612757E-04
CEMTR= 0.29881073E-04
CBASE= 0.26831662E-06

TIME= 0.84499945E-04
VIN= 0.12000000E 02
VOUT= 0.47278304E 01
VCLTR= 0.73912325E 01
VBASE= 0.49343734E 01
VBE= 0.20454311E-00
VCE= 0.24568590E 01
CURL= 0.59158475E-02
CCLTR= 0.28414980E-08
CEMTR= 0.59252533E-09
CBASE= -0.22489726E-08

TIME= 0.84999945E-04
VIN= 0.12000000E 02
VOUT= 0.52976867E 01
VCLTR= 0.90608209E 01
VBASE= 0.49880759E 01
VBE= -0.30961072E-00
VCE= 0.40727449E 01
CURL= 0.61047394E-02
CCLTR= 0.40728921E-08
CEMTR= -0.30961205E-09
CBASE= -0.43825042E-08

TIME= 0.85499944E-04
VIN= 0.12000000E 02
VOUT= 0.58754417E 01
VCLTR= 0.10767710E 02
VBASE= 0.49011418E 01
VBE= -0.97429984E 00
VCE= 0.58665682E 01
CURL= 0.62091348E-02
CCLTR= 0.58667155E-08
CEMTR= -0.97430117E-09
CBASE= -0.68410166E-08

TIME= 0.85999943E-04
VIN= 0.12000000E 02
VOUT= 0.64553985E 01
VCLTR= 0.12488896E 02
VBASE= 0.47662752E 01
VBE= -0.16891232E 01
VCE= 0.77226211E 01
CURL= 0.62277292E-02
CCLTR= 0.77227683E-08
CEMTR= -0.16891245E-08
CBASE= -0.94118928E-08

TIME= 0.86499943E-04
VIN= 0.12000000E 02
VOUT= 0.70286874E 01
VCLTR= 0.14199811E 02
VBASE= 0.46379902E 01

VBE= -0.23906972E 01
VCE= 0.95618209E 01
CURL= 0.61604157E-02
CCLTR= 0.95619681E-08
CEMTR= -0.23906985E-08
CBASE= -0.11952666E-07

TIME= 0.86999942E-04
VIN= 0.12000000E 02
VOUT= 0.75860901E 01
VCLTR= 0.15875419E 02
VBASE= 0.45310041E 01
VBE= -0.30550859E 01
VCE= 0.11344415E 02
CURL= 0.60083359E-02
CCLTR= 0.11344562E-07
CEMTR= -0.30550873E-08
CBASE= -0.14399649E-07

TIME= 0.87499941E-04
VIN= 0.12000000E 02
VOUT= 0.81186147E 01
VCLTR= 0.17491178E 02
VBASE= 0.44419126E 01
VBE= -0.36767021E 01
VCE= 0.13049266E 02
CURL= 0.57738718E-02
CCLTR= 0.13049413E-07
CEMTR= -0.36767034E-08
CBASE= -0.16726116E-07

TIME= 0.87999940E-04
VIN= 0.12000000E 02
VOUT= 0.86177646E 01
VCLTR= 0.19023608E 02
VBASE= 0.43634199E 01
VBE= -0.42543446E 01
VCE= 0.14660188E 02
CURL= 0.54606090E-02
CCLTR= 0.14660335E-07
CEMTR= -0.42543460E-08
CBASE= -0.18914681E-07

TIME= 0.88499939E-04
VIN= 0.12000000E 02
VOUT= 0.90756722E 01
VCLTR= 0.20450651E 02
VBASE= 0.42904339E 01
VBE= -0.47852383E 01
VCE= 0.16160217E 02
CURL= 0.50732698E-02
CCLTR= 0.16160364E-07
CEMTR= -0.47852396E-08
CBASE= -0.20945603E-07

TIME= 0.88999939E-04
VIN= 0.12000000E 02
VOUT= 0.94851704E 01
VCLTR= 0.21751948E 02
VBASE= 0.42200430E 01
VBE= -0.52651274E 01
VCE= 0.17531905E 02
CURL= 0.46176413E-02
CCLTR= 0.17532051E-07
CEMTR= -0.52651288E-08
CBASE= -0.22797181E-07

TIME= 0.89499938E-04
VIN= 0.12000000E 02
VOUT= 0.98399273E 01
VCLTR= 0.22909169E 02
VBASE= 0.41565531E 01
VBE= -0.56833740E 01
VCE= 0.18752615E 02
CURL= 0.41004837E-02
CCLTR= 0.18752763E-07
CEMTR= -0.56833754E-08
CHASE= -0.24436138E-07

TIME= 0.89999937E-04
VIN= 0.12000000E 02
VOUT= 0.10134432E 02
VCLTR= 0.23906133E 02
VBASE= 0.41033109E 01
VBE= -0.60311207E 01
VCE= 0.19802822E 02
CURL= 0.35294193E-02
CCLTR= 0.19802970E-07
CEMTR= -0.60311221E-08
CHASE= -0.25834092E-07

TIME= 0.90499937E-04
VIN= 0.12000000E 02
VOUT= 0.10363684E 02
VCLTR= 0.24728481E 02
VBASE= 0.40236661E 01
VBE= -0.63400182E 01
VCE= 0.20704816E 02
CURL= 0.29128096E-02
CCLTR= 0.20704962E-07
CEMTR= -0.63400196E-08
CHASE= -0.27044982E-07

TIME= 0.90999936E-04
VIN= 0.12000000E 02
VOUT= 0.10524327E 02
VCLTR= 0.25365391E 02
VBASE= 0.39470548E 01
VBE= -0.65772731E 01
VCE= 0.21418337E 02
CURL= 0.22596701E-02
CCLTR= 0.21418484E-07
CEMTR= -0.65772744E-08
CHASE= -0.27995758E-07

TIME= 0.91499935E-04
VIN= 0.12000000E 02
VOUT= 0.10613853E 02
VCLTR= 0.25808528E 02
VBASE= 0.38868617E 01
VBE= -0.67269912E 01
VCE= 0.21921666E 02
CURL= 0.15795077E-02
CCLTR= 0.21921813E-07
CEMTR= -0.67269925E-08
CHASE= -0.28648806E-07

TIME= 0.91999935E-04
VIN= 0.12000000E 02
VOUT= 0.10630318E 02
VCLTR= 0.26051704E 02
VBASE= 0.38074101E 01

VBE= -0.68229078E 01
 VCE= 0.22244294E 02
 CURL= 0.88216111E-03
 CCLTR= 0.22244441E-07
 CEMTR= -0.68229091E-08
 CBASE= -0.29067350E-07

TIME= 0.92499933E-04
 VIN= 0.12000000E 02
 VOUT= 0.10573659E 02
 VCLTR= 0.26092804E 02
 VBASE= 0.37554033E 01
 VBE= -0.68182562E 01
 VCE= 0.22337401E 02
 CURL= 0.17771453E-03
 CCLTR= 0.22337548E-07
 CEMTR= -0.68182576E-08
 CBASE= -0.29155806E-07

TIME= 0.92999933E-04
 VIN= 0.12000000E 02
 VOUT= 0.10443962E 02
 VCLTR= 0.25931326E 02
 VBASE= 0.36827850E 01
 VBE= -0.67611773E 01
 VCE= 0.22248541E 02
 CURL= -0.52372052E-03
 CCLTR= 0.22248688E-07
 CEMTR= -0.67611787E-08
 CBASE= -0.29009866E-07

TIME= 0.93499932E-04
 VIN= 0.12000000E 02
 VOUT= 0.10242997E 02
 VCLTR= 0.25570590E 02
 VBASE= 0.36057527E 01
 VBE= -0.66372444E 01
 VCE= 0.21964837E 02
 CURL= -0.12120894E-02
 CCLTR= 0.21964984E-07
 CEMTR= -0.66372456E-08
 CBASE= -0.28602230E-07

TIME= 0.93999931E-04
 VIN= 0.12000000E 02
 VOUT= 0.99735647E 01
 VCLTR= 0.25016785E 02
 VBASE= 0.35401003E 01
 VBE= -0.64334583E 01
 VCE= 0.21476678E 02
 CURL= -0.18775637E-02
 CCLTR= 0.21476825E-07
 CEMTR= -0.64334597E-08
 CBASE= -0.27710285E-07

TIME= 0.94499931E-04
 VIN= 0.12000000E 02
 VOUT= 0.96392060E 01
 VCLTR= 0.24278509E 02
 VBASE= 0.34760698E 01
 VBE= -0.61631361E 01
 VCE= 0.20807439E 02
 CURL= -0.25106928E-02
 CCLTR= 0.20802586E-07
 CEMTR= -0.61631375E-08
 CBASE= -0.26965723E-07

TIME= 0.94999930E-04
VIN= 0.12000000E 02
VOUT= 0.92444876E 01
VCLTR= 0.23367132E 02
VBASE= 0.34132338E 01
VBE= -0.58312537E 01
VCE= 0.19953898E 02
CURL= -0.31025279E-02
CCLTR= 0.19954044E-07
CEMTR= -0.58312550E-08
CBASE= -0.25785299E-07

TIME= 0.95499929E-04
VIN= 0.12000000E 02
VOUT= 0.87948431E 01
VCLTR= 0.22296483E 02
VBASE= 0.33514965E 01
VBE= -0.54433466E 01
VCE= 0.18944987E 02
CURL= -0.36447497E-02
CCLTR= 0.18945134E-07
CEMTR= -0.54433480E-08
CBASE= -0.24388482E-07

TIME= 0.95999929E-04
VIN= 0.12000000E 02
VOUT= 0.82964919E 01
VCLTR= 0.21082652E 02
VBASE= 0.32909531E 01
VBE= -0.50055388E 01
VCE= 0.17791700E 02
CURL= -0.41297876E-02
CCLTR= 0.17791846E-07
CEMTR= -0.50055401E-08
CBASE= -0.22797386E-07

TIME= 0.96499927E-04
VIN= 0.12000000E 02
VOUT= 0.77563442E 01
VCLTR= 0.19743737E 02
VBASE= 0.32317554E 01
VBE= -0.45245887E 01
VCE= 0.16511981E 02
CURL= -0.45509274E-02
CCLTR= 0.16512129E-07
CEMTR= -0.45245901E-08
CBASE= -0.21036719E-07

TIME= 0.96999927E-04
VIN= 0.12000000E 02
VOUT= 0.71818917E 01
VCLTR= 0.18299567E 02
VBASE= 0.31733199E 01
VBE= -0.40085717E 01
VCE= 0.15126247E 02
CURL= -0.49024048E-02
CCLTR= 0.15126394E-07
CEMTR= -0.40085731E-08
CBASE= -0.19134967E-07

TIME= 0.97499926E-04
VIN= 0.12000000E 02
VOUT= 0.65811287E 01
VCLTR= 0.16771454E 02
VBASE= 0.31169826E 01

VBE= -0.34641460E 01
VCE= 0.13654471E 02
CURL= -0.51794842E-02
CCLTR= 0.13654618E-07
CEMTR= -0.34641474E-08
CBASE= -0.17118765E-07

TIME= 0.97999925E-04
VIN= 0.12000000E 02
VOUT= 0.59623683E 01
VCLTR= 0.15181784E 02
VBASE= 0.30600117E 01
VBE= -0.29023566E 01
VCE= 0.12121772E 02
CURL= -0.53785232E-02
CCLTR= 0.12121919E-07
CEMTR= -0.29023570E-08
CBASE= -0.15024277E-07

TIME= 0.98499925E-04
VIN= 0.12000000E 02
VOUT= 0.53342298E 01
VCLTR= 0.13553838E 02
VBASE= 0.30642617E 01
VBE= -0.23299681E 01
VCE= 0.10549577E 02
CURL= -0.54970230E-02
CCLTR= 0.10549724E-07
CEMTR= -0.23299694E-08
CBASE= -0.12879694E-07

TIME= 0.98999924E-04
VIN= 0.12000000E 02
VOUT= 0.47054151E 01
VCLTR= 0.11911318E 02
VBASE= 0.29474615E 01
VBE= -0.17579536E 01
VCE= 0.89638568E 01
CURL= -0.55336604E-02
CCLTR= 0.89640039E-08
CEMTR= -0.17579550E-08
CBASE= -0.10721958E-07

TIME= 0.99499923E-04
VIN= 0.12000000E 02
VOUT= 0.40845878E 01
VCLTR= 0.10277999E 02
VBASE= 0.28839583E 01
VBE= -0.12006294E 01
VCE= 0.73940413E 01
CURL= -0.54883072E-02
CCLTR= 0.73941885E-08
CEMTR= -0.12006307E-08
CBASE= -0.85948192E-08

TIME= 0.99999923E-04
VIN= 0.12000000E 02
VOUT= 0.34804593E 01
VCLTR= 0.86776680E 01
VBASE= 0.28156487E 01
VBE= -0.66481057E 00
VCE= 0.58620193E 01
CURL= -0.53620147E-02
CCLTR= 0.58621666E-08
CEMTR= -0.66481189E-09
CBASE= -0.65269784E-08

TIME= 0.10049992E-03
VIN= 0.12000000E 02
VOUT= 0.29014388E 01
VCLTR= 0.71334498E 01
VRASE= 0.27345600E 01
VBE= -0.16687874E-00
VCE= 0.43988896E 01
CURL= -0.51570117E-02
CCLTR= 0.43990371E-08
CENTR= -0.16687983E-09
CBASE= -0.45659170E-08

FOUT = 2

COMPUTER GENERATED

OUTPUT CURVES

SC4020 PLOTTER

11.00000000

8.00000000

V
O
U
T

V
O
L
S

5.00000000

2.00000000

-1.00000000

0.00000000

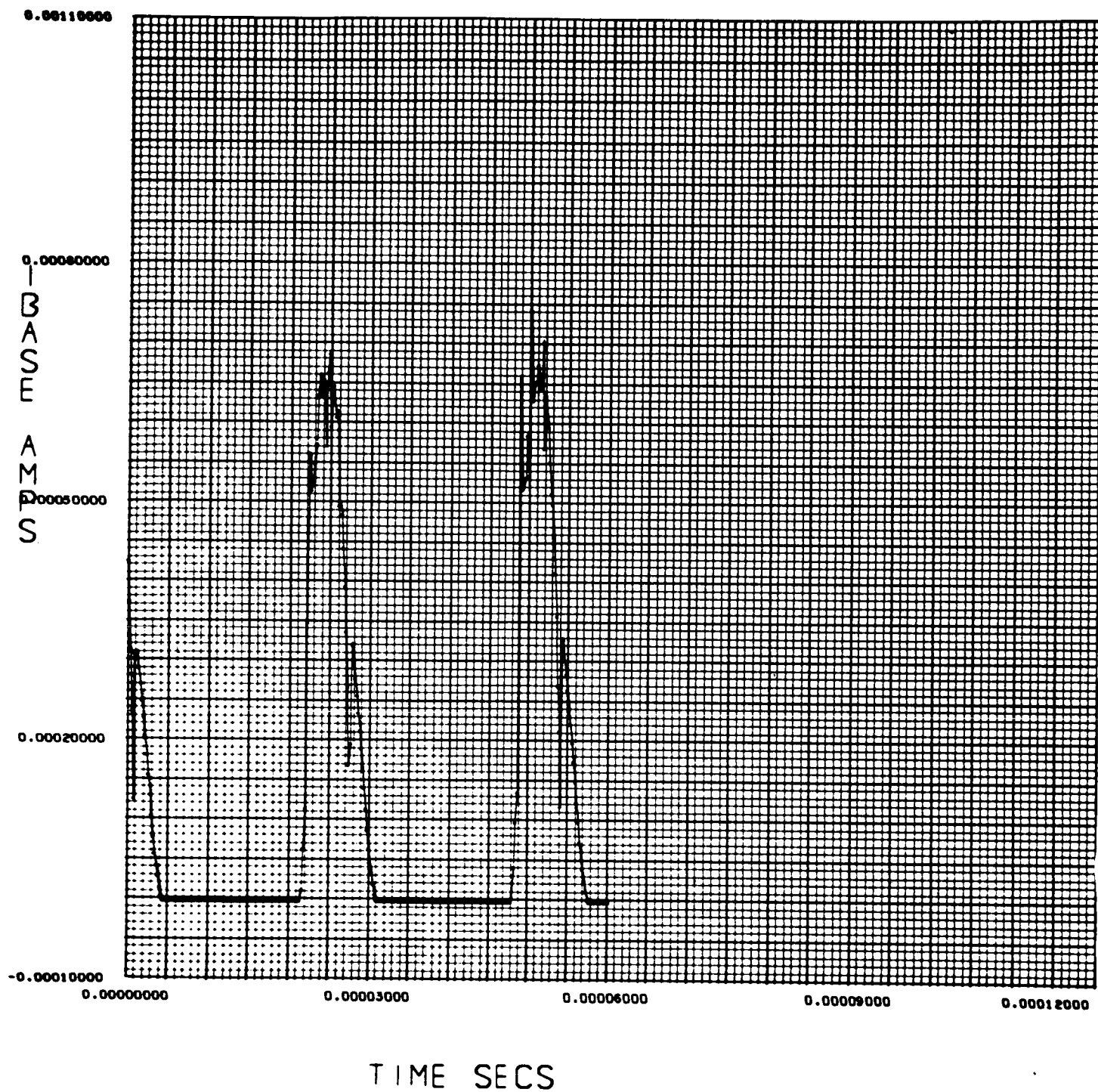
0.0003000

0.0006000

0.0009000

0.0012000

T I M E S E C S



34.99990000

39.99990000

V
C
B
V
O
L
T
S

29.99990000

9.99990000

-4.99990000

0.00000000

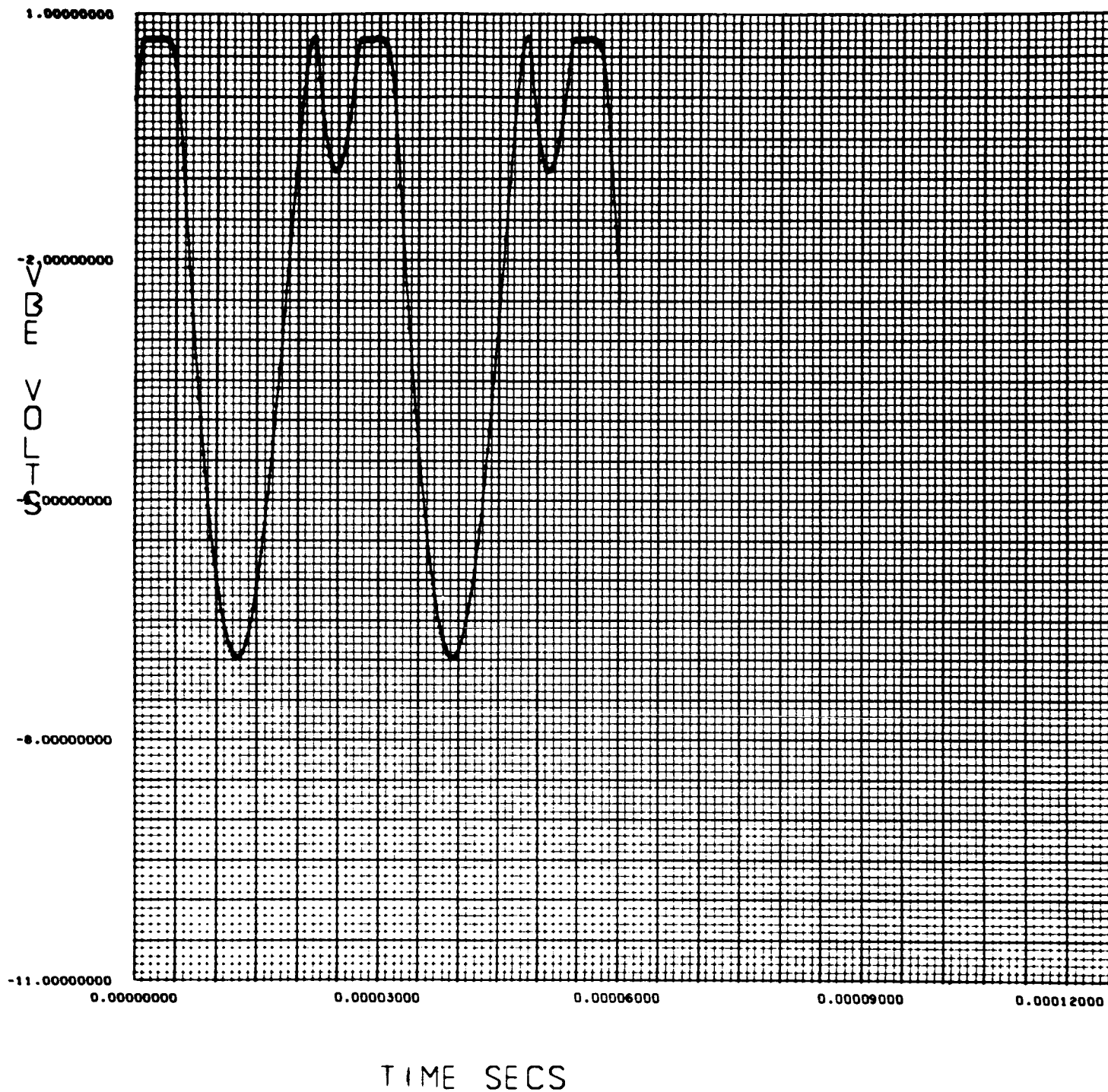
0.00003000

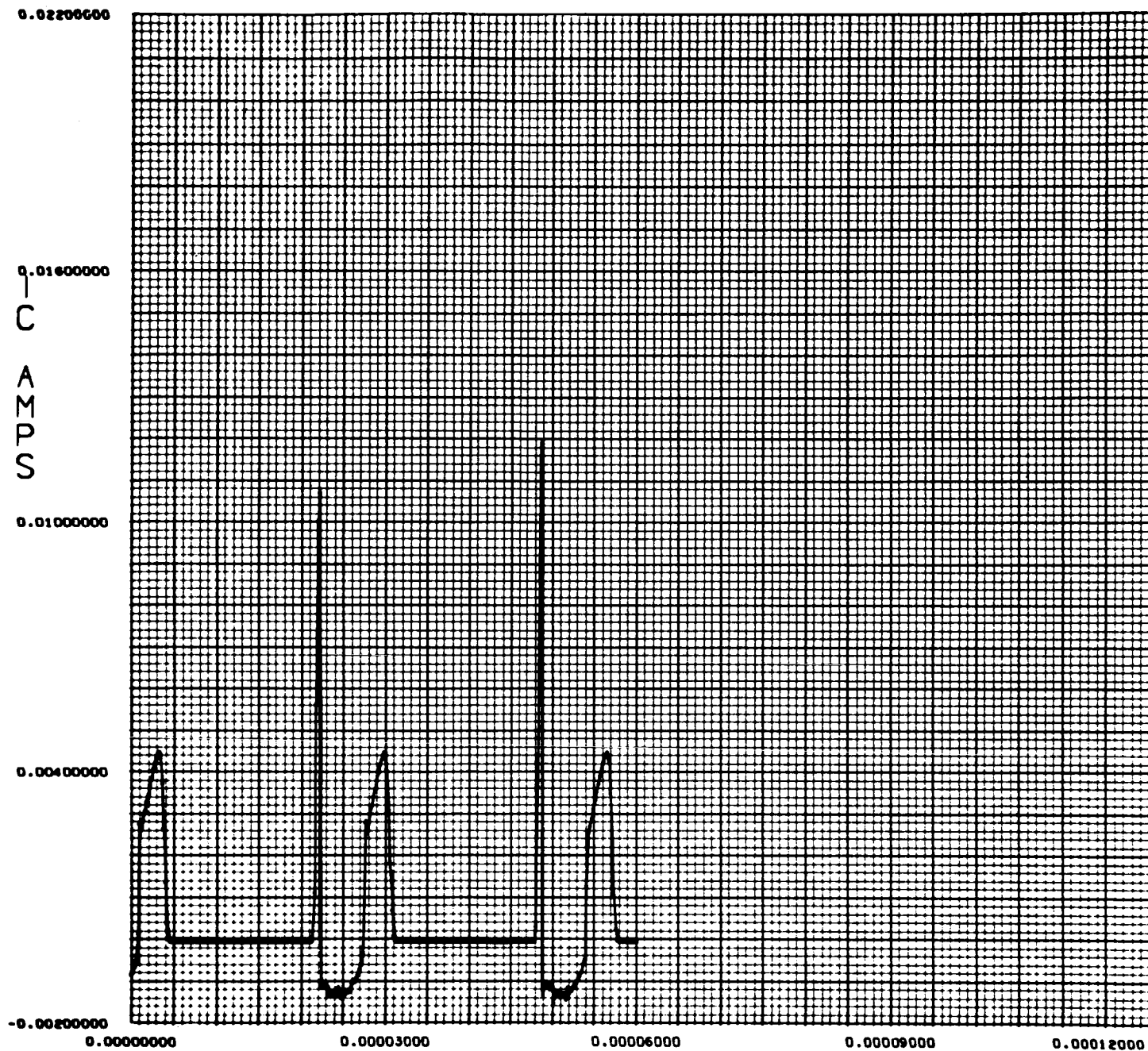
0.00006000

0.00009000

0.00012000

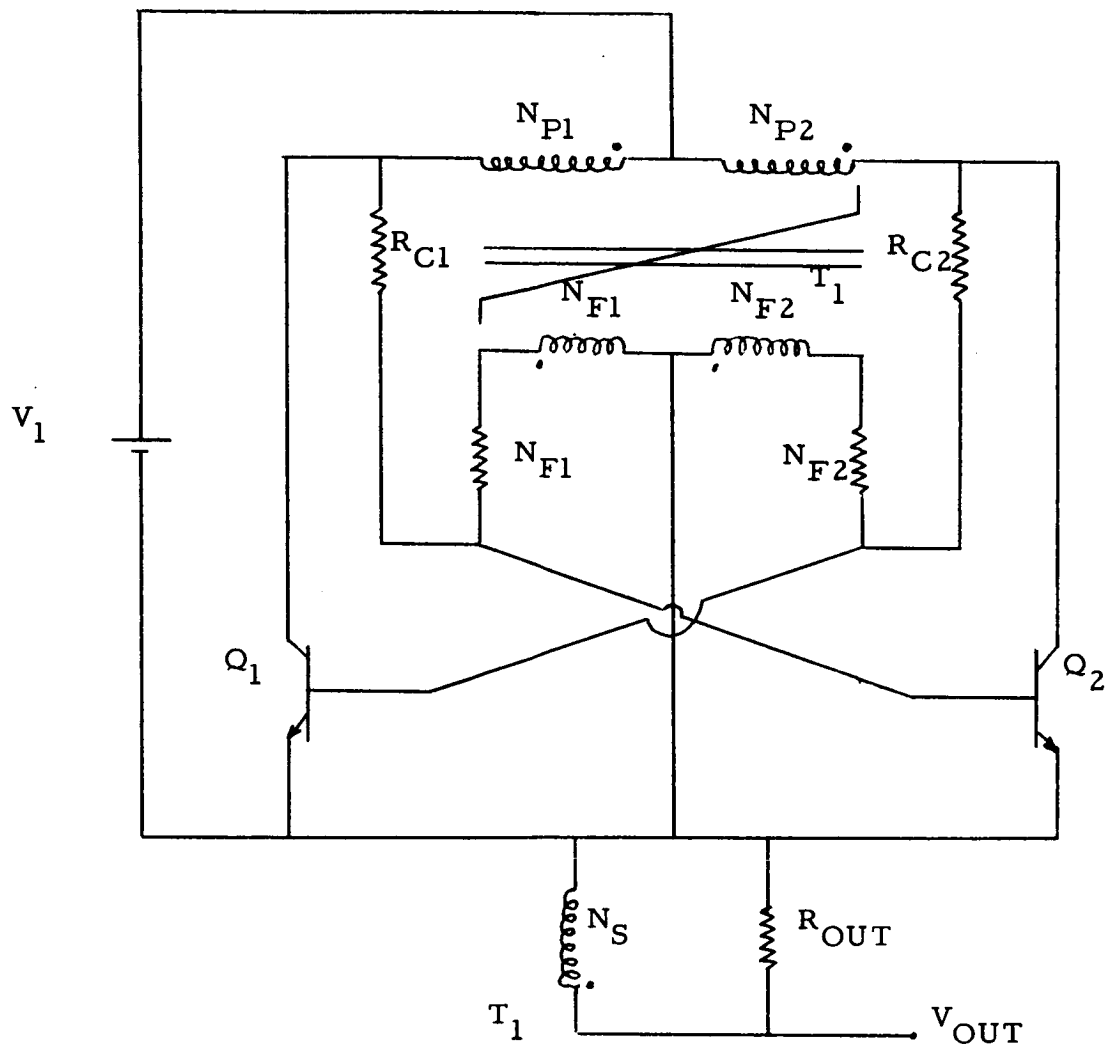
TIME SECS



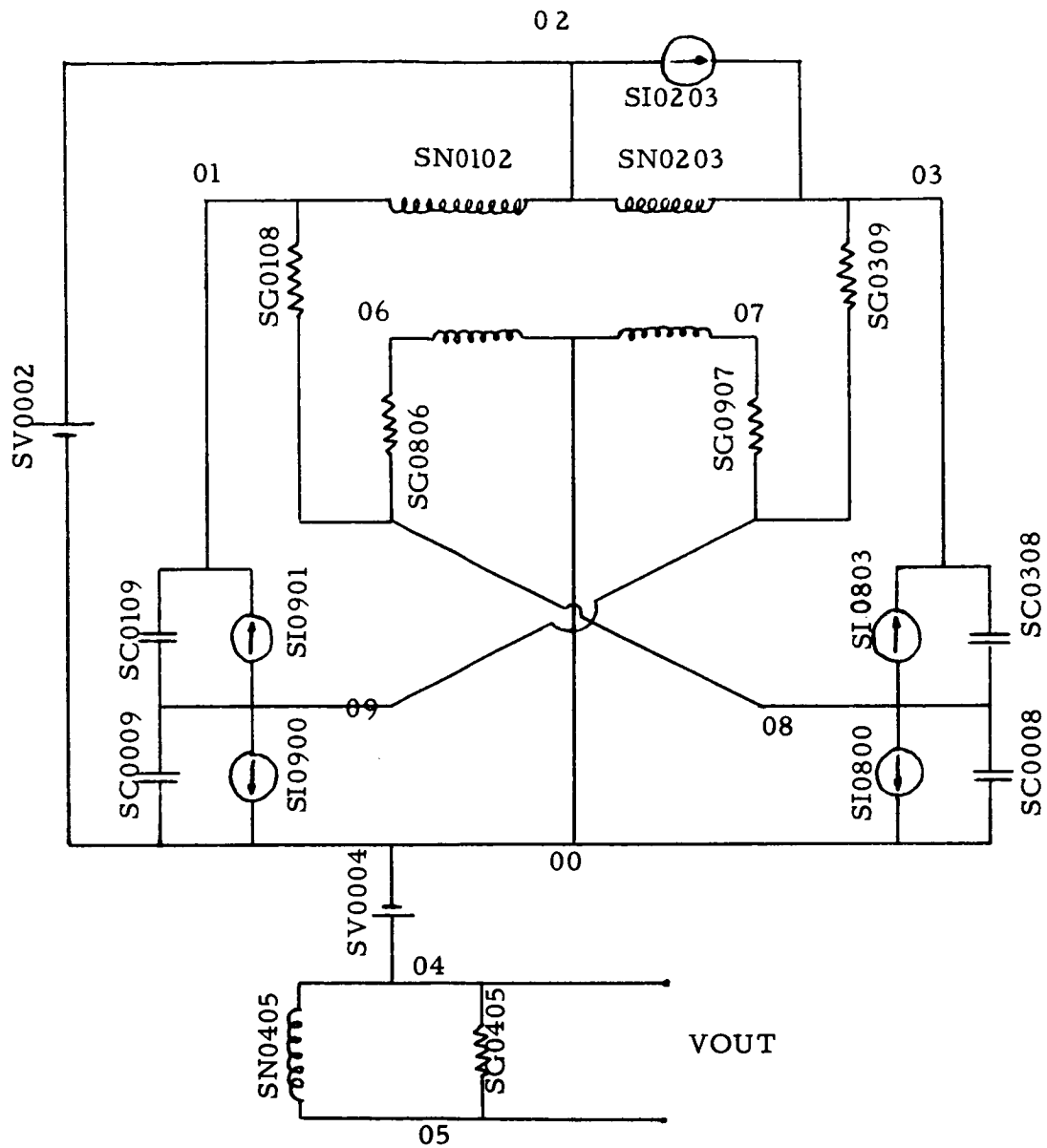


C. Example Flux Oscillator

1. Problem Preparation



Flux Oscillator
Original Circuit Schematic



Flux Oscillator
TAG Equivalent Circuit

2. Program Listings

Computer Listing
TAG Circuit Description
and
Data Decks for Two Cases

CIRCUIT DESCRIPTION

SV0002 , SV0004 ,

SC0109 , SC0308 ,

SC0009 , SC0008 ,

SG0405 , SG0806 , SG0907 , SG0108 , SG0309 ,

SI0900 , SI0901 ,

SI0800 , SI0803 ,

SI0203 ,

SN0600/01-1 ,

SN0007/01-1 ,

SN0102/01-4 ,

SN0203/01-4 ,

SN0405/01-2 *

NUMBER OF NODES IS 9

2SC 55636 MAX 698 COUNT 22

7

6

4

3

2

SV0002 = VP20

CALL TRAN3(SV0009,SV0109,\$SI0900,\$SI0901,\$SC0009,\$SC0109,DATAQ1,

1DUM,DUM,LALGFT,0)

CALL TRAN3(SV0008,SV0308,\$SI0800,\$SI0803,\$SC0008,\$SC0308,DATAQ2,

1DUM,DUM,LALGFT,0)

CALL PLIND (SS0302,RIND1,STATE1,CISAT1,\$1FLXST1,\$2FLXST2,

1FLXIN1,DATA1,LLCNT,LALGFT,1)

SI0203 = RIND1*(SS0302+FLXIN1) -CISAT1*STATE1

DIMENSION BE1(100),BF2(100),BF3(100),BF4(100),BF5(100),BF6(100)

DIMENSION DATA1(6), DATAQ1(17), DATAQ2(17)

30 CONTINUE

TIME = FT

STEP = FHC

VOUT = SV0405

VBQ1 = SV0009

VBQ2 = SV0008

VCQ1 = SV0001

VCQ2 = SV0003

VFB1 = SV0007

VFB2 = SV0006

FIMAG = SI0203

VPR1 = SV0302

PFLUX = SS0302 +FLXIN1

STATE = STATE1

RIND = RIND1

J=1

IF(FT-DONE) 60,60,61

61 J=2

60 CALL SCOPE(FT,VOUT,BF1,100,J,10HVOUT VOLTS,9HTIME SECS)

CALL SCOPE(FT,VBQ1,BF2,100,J,10HVBQ1 VOLTS,9HTIME SECS)

CALL SCOPE(FT,VCQ1,BF3,100,J,10HVCQ1 VOLTS,9HTIME SECS)

CALL SCOPE(FT,VFB1,BF4,100,J,10HVFB1 VOLTS,9HTIME SECS)

CALL SCOPE(FIMAG,PFLUX,BF5,100,J,14HFLUX VOLT-SECS,9HIMAG AMPS)

CALL SCOPE (FT,FIMAG,BF6,100,J,9HIMAG AMPS,9HTIME SECS)

IF(J-1)6200,6200,6000

END

SV0002 = VP20

CALL TRAN3(SV0009,SV0109,\$SI0900,\$SI0901,\$SC0009,\$SC0109,DATAQ1,
1DUM,DUM,LALGFT,0)CALL TRAN3(SV0008,SV0308,\$SI0800,\$SI0803,\$SC0008,\$SC0308,DATAQ2,
1DUM,DUM,LALGFT,0)CALL PLIND (SS0302,RIND1,STATE1,CISAT1,\$1FLXST1,\$2FLXST2,
1FLXIN1,DATA1,LLCNT,LALGFT,1)

SI0203 = RIND1*(SS0302+FLXIN1) -CISAT1*STATE1

DIMENSION BF1(100),BF2(100),BF3(100),BF4(100),BF5(100),BF6(100)

DIMENSION DATA1(6), DATAQ1(17), DATAQ2(17)

30 CONTINUE

TIME = FT

STEP = FHC

VOUT = SV0405

VBQ1 = SV0009

VBQ2 = SV0008

VCQ1 = SV0001

VCQ2 = SV0003

VFB1 = SV0007

VFB2 = SV0006

FIMAG = SI0203

VPR1 = SV0302

PFLUX = SS0302 +FLXIN1

STATE = STATE1

RIND = RIND1

J=1

IF(FT-DONE) 60,60,61

61 J=2

60 CALL SCOPE(FT,VOUT,BF1,100,J,10HVOUT VOLTS,9HTIME SECS)

CALL SCOPE(FT,VBQ1,BF2,100,J,10HVBQ1 VOLTS,9HTIME SECS)

CALL SCOPE(FT,VCQ1,BF3,100,J,10HVCQ1 VOLTS,9HTIME SECS)

CALL SCOPE(FT,VFB1,BF4,100,J,10HVFB1 VOLTS,9HTIME SECS)

CALL SCOPE(FIMAG,PFLUX,BF5,100,J,14HFLUX VOLT-SECS,9HIMAG AMPS)

CALL SCOPE (FT,FIMAG,BF6,100,J,9HIMAG AMPS,9HTIME SECS)

IF(J-1)6200,6200,6000

END

SETHI	45622		
SETLOW	45617		
WRITEB	45407		
WRITED	45405		
READB	45402		
READC	45400		
(UNIT)	45616		
ENDFIL	45641		
(IOU)	46253	00000	46250
PDUMP	46305	00003	46301
DUMP	46304		
XMIN	46530	00000	46530
MINUTE	46530		
CLOCK	46540		

UNUSED CORE 46574 THRU 71571

EXECUTION 002919

THIS RUN IS INVERTER 3 USING PLIND FOR THE PIECEWISE LINEAR
INDUCTOR MODEL, U RATIO OF 1000.

VP2C = 20.0 ,

SG0405 = .390 E-2 ,

SG0806 = .464 E-3 ,

SG0907 = .464 E-3 ,

SG0108 = .375 E-4 ,

SG0309 = .375 E-4 ,

DATAQ1 = .750E-13,1.49E-13,1.E-9,1.E-9,.9910,.250,.199E-8,.100E-7,

.750,.750,.50,.33,.095E-9,.033E-9,0.0,0.0,.026,

DATAQ2 = .750E-13,1.49E-13,1.E-9,1.E-9,.9901,.250,.199E-8,.100E-7,

.750,.750,.50,.33,.095E-9,.033E-9,0.0,0.0,.026,

DATA1 = 100., 2.75, .0005, 15.E3, .333E6, 1000.,

FEPSL2 = 5.E-8 ,

FEPSL3 = 5.E-6 ,

FOUT = 0.1E-6, DONE = 20.0E-6*

SAME AS ABOVE BUT THE U RATIO IS CHANGED TO 500.
DATA1(6) = 500.*

7

6

5

4

3

2

Computer Generated Output Curves
SC 40 20 Plotter

W. J. THOMAS

DATE 05/25/66

TIME 000501

11-2

29.99990000

14.99990000

VOLTS

-4.99990000

-14.99990000

-29.99990000

0.00000000

0.00000000

0.00001000

0.00001000

0.00002000

TIME SECS

99.99999999

99.99999999

VCO1

VCO2

31.99999999

9.99999999

-19.99999999

0.00000000

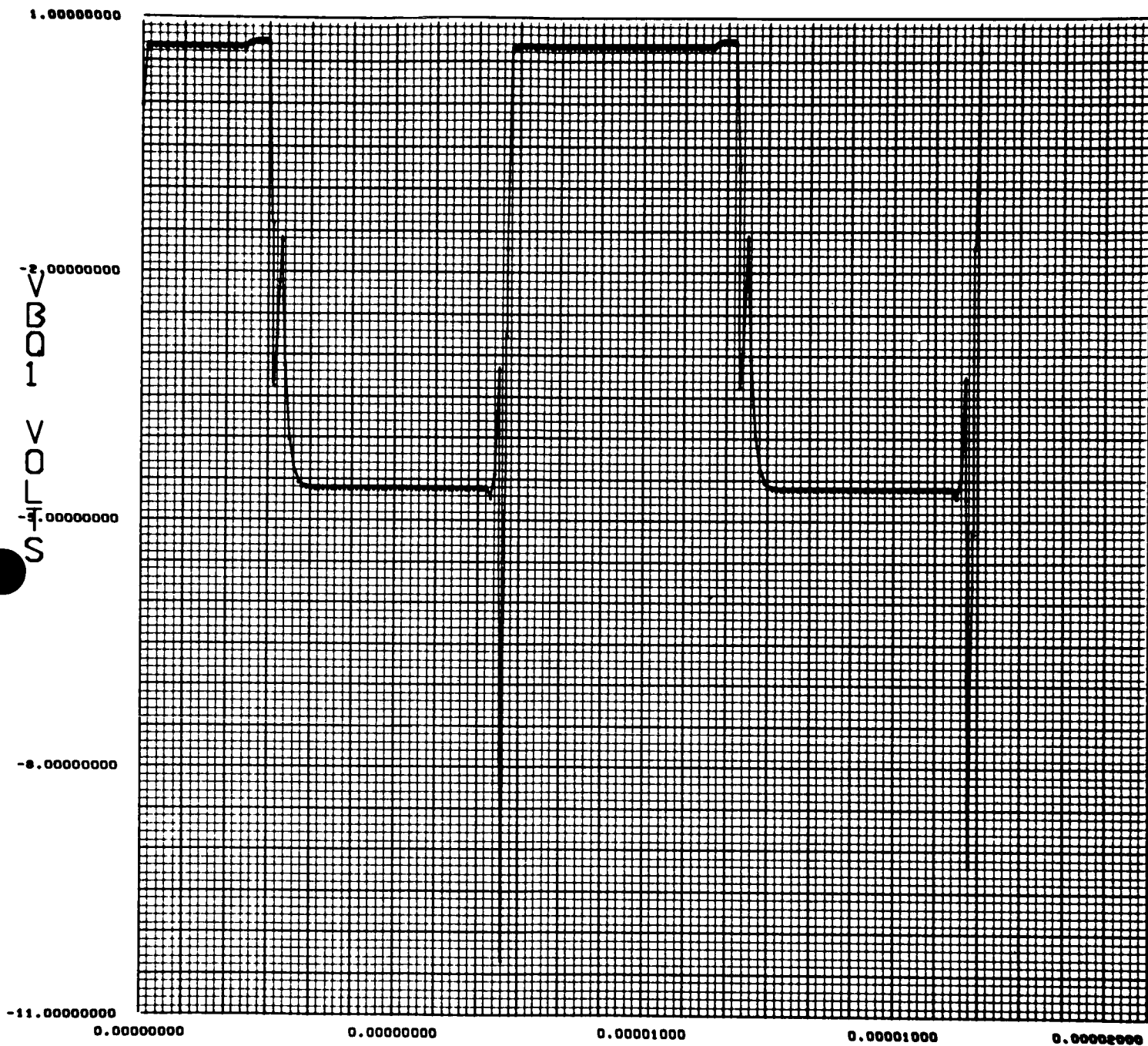
0.00000000

0.00001000

0.00001000

0.00002000

TIME SECS



TIME SECS

11.99999999

V
F
B
I
V
O
L
T
S

-5.99999999

-11.99999999

0.00000000

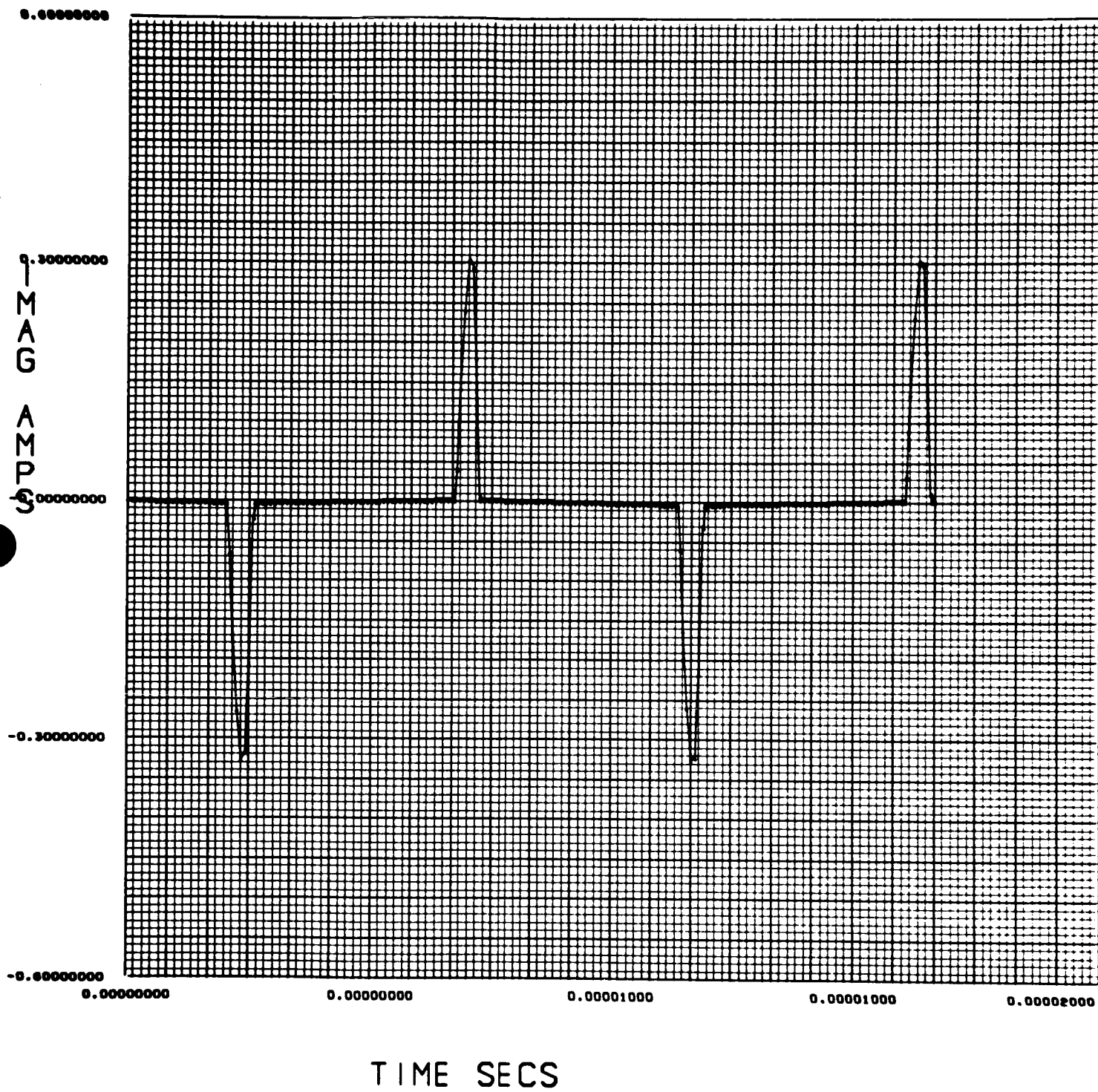
0.00000000

0.00001000

0.00001000

0.00000000

TIME SECS



0.00000000

0.0003000

0.0000000

-0.0003000

-0.0006000

-0.60000000

-0.30000000

-0.00000000

0.30000000

0.60000000

IMAG AMPS

49.99990000

19.99990000

VOLTS

9.99990000

-39.99990000

-59.99990000

0.00000000

0.00000000

0.00001000

0.00001000

0.00002000

TIME SECS

179.99999999

119.99999999

V
C
O
I
V
O
L
T
S

59.99999999

-0.00000000

-59.99999999

0.00000000

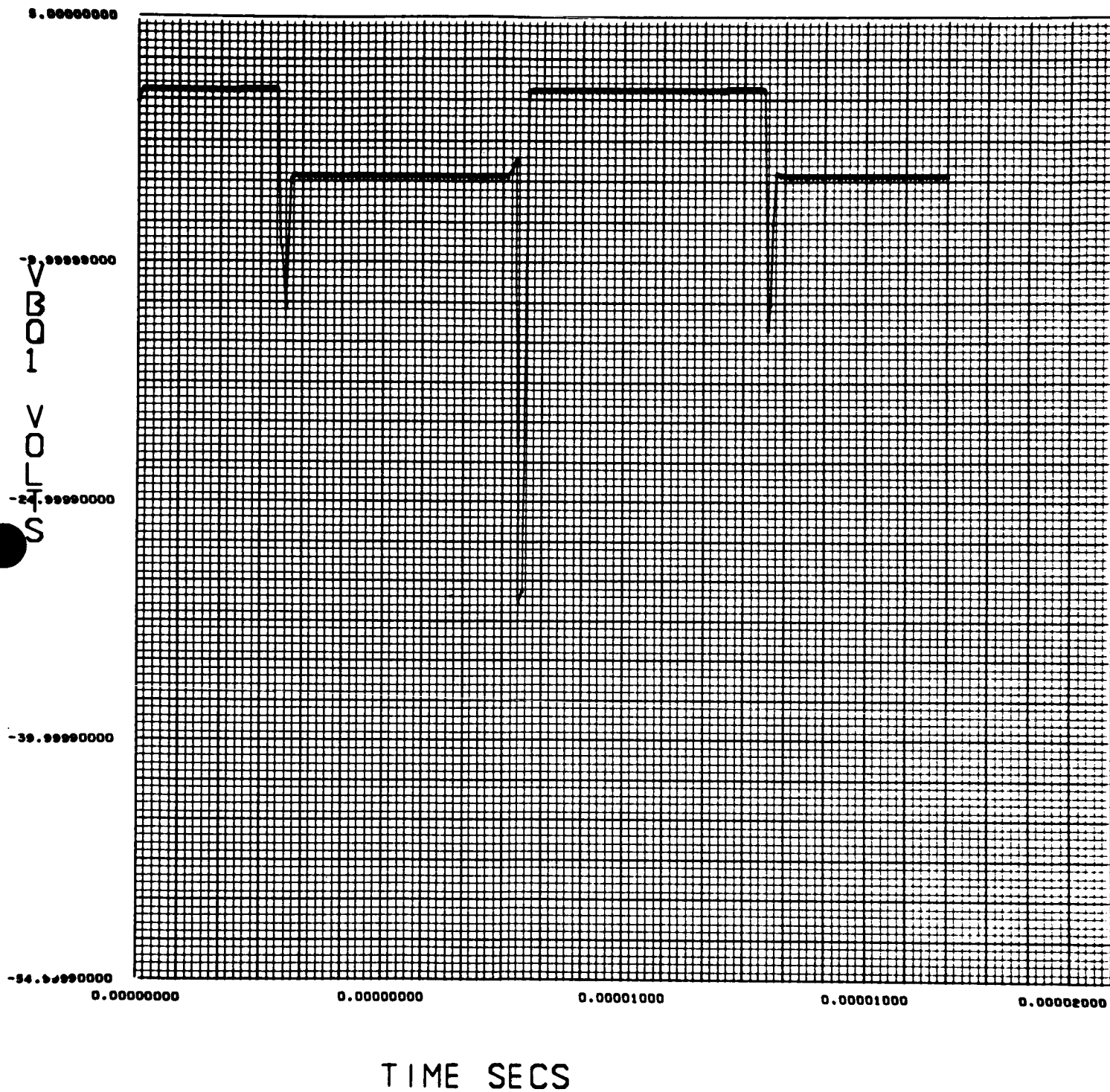
0.00000000

0.00001000

0.00001000

0.00002000

TIME SECS



34.99990000

V
T
B
I
V
O
L
T
S

-19.99990000

-34.99990000

0.00000000

0.00000000

0.00001000

0.00001000

0.00002000

TIME SECS



TIME SECS

